Problèmes de plus courts chemins dans les NoC et leurs extensions aux cas difficiles

Boureima Zerbo

THESE / UNIVERSITÉ DE BRETAGNE-SUD
sous le sceau de l'Université européenne de Bretagne

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE-SUD

Mention : STIC

École doctorale SICMA

présenté par

Boureima ZERBO

Laboratoire des Sciences et Technologies de l'Information, de la Communication et de la Connaissance

THESE / UNIVERSITÉ DE OUAGADOUGOU

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE OUAGADOUGOU

Option Sciences Appliquées – Spécialité Informatique

École doctorale Sciences et Technologies

Laboratoire de Traitement de l'Information et de la Communication

Problèmes de plus courts chemins dans les NoC et leurs extensions aux cas difficiles

Thèse soutenue le 03 décembre 2012,
devant la commission d'examen composée de :

M. Aziz Moukrim
Professeur des Universités, Université de Technologie de Compiègne / Président

M. Lionel Amodéo
Professeur des Universités, Université de Technologie de Troyes / Rapporteur

M. Philippe Lacomme
Maître de Conférences HDR, Université Blaise Pascal, Clermont-Ferrand / Rapporteur

M. Claude Yugma
Chargé de Recherche, École des Mines de St Étienne / Examinateur

M. Jean-Philippe Diguet
Directeur de Recherche, CNRS, Lab-STICC, Université de Bretagne-Sud, Lorient / Invité

M. Marc Sevaux
Professeur des Universités, Lab-STICC, Université de Bretagne-Sud, Lorient / Directeur de thèse

M. Oumarou Sié
Maître de Conférences CAMES, LTIC, Université de Ouagadougou, Burkina Faso / Directeur de thèse

M. André Rossi
Maître de Conférences HDR, Lab-STICC, Université de Bretagne-Sud, Lorient / Co-Directeur de thèse

M. Jean-Charles Créput
Maître de Conférences HDR, Université de Technologie de Belfort-Montbéliard / Co-Directeur de thèse

Préambule

Le financement des différents travaux et séjours de recherche ayant conduit au présent manuscrit de thèse, a été possible grâce à l'engagement soutenu des institutions suivantes:

- **Le Programme de bourses Robert S. McNamara (RSM) de la Banque Mondiale**, qui a financé les six derniers mois (juillet à décembre 2012) de mes travaux dans mon laboratoire d'accueil (Lab-STICC) à Lorient et la rédaction du manuscrit;

- **Le Projet RÉSEAU** de la **Coopération Universitaire Franco-Burkinabé**, pour le financement de mon premier (mars à mais 2008), deuxième (mai 2009) et cinquième (avril à juin 2012) voyage de recherche;

- **Le Programme *Application-Field-Aware Adaptive Network on chip Architecture* (AFANA) ou "Architecture du futur"-2007/2010 de l'Agence Nationale de la Recherche (ANR) de France** pour le financement de mon troisième (mars à septembre 2010) voyage de recherche;

- **Le Fonds National pour l'Éducation et la Recherche (FONER) du Burkina** qui a contribué au financement de mon quatrième (décembre 2011 à mars 2012) voyage de recherche;

- **L'Université de Bretagne Sud (UBS)**, en plus de son environnement humain, intellectuel et matériel m'a offert sept mois (décembre 2011 à juin 2012) de loyer en résidence universitaire;

- **L'Université Ouaga 2 (UO2)**, mon employeur, qui a pris en charge les frais de transport de mon quatrième séjour de recherche, et le coût inestimable d'une libération de mes obligations professionnelles durant mes voyages de recherche.

Résumé

Nous définissons et étudions un problème d'optimisation combinatoire et un programme linéaire en nombres entiers, qui modélisent le routage multi-chemin avec la qualité de service garantie de trafic dans un réseau sur puce. Basé sur le multiplexage temporel et l'émission cyclique des messages, le modèle permet d'éviter les collisions, les blocages statiques et dynamiques dans des réseaux à topologie irrégulière, tout en minimisant les temps de latence. Une extension de ce problème de routage multi-chemin, qui permet une reconfiguration dynamique du routage au moment de l'exécution est également présentée. Dans ce cas, des ensembles indépendants de chemins valides sont pré-calculés de telle sorte qu'ils peuvent être inter-changés en cours d'exécution sans impact sur le trafic courant, tout en réutilisant tous les intervalles de temps dont les ressources sont vacantes ou libérées.
L'approche du graphe spatio-temporel étendu est retenue dans les processus de résolution. Tout d'abord, nous présentons un ensemble d'opérateurs de base de calcul de plus courts chemins. Se sont une heuristique de construction parallèle gloutonne, un opérateur de voisinage, et un algorithme de Dijkstra modifié dans un graphe spatio-temporel étendu qui calcul un chemin unique dans un NoC occupé en temps pseudo-polynomial. Ensuite, pour résoudre l'ensemble des problèmes, les opérateurs sont introduits et combinés dans trois méthodes de recherche locale itérée capable de générer rapidement des solutions admissibles, un algorithme évolutionnaire à base de population solutions conférant une grande diversité à la recherche de solutions et un algorithme mémétique, tirant partie des avantages des deux précédents.
Les expériences sont réalisées sur un ensemble d'instances d'applications réelles, et d'instances d'applications artificielles qui sont générées aléatoirement à partir des instances réelles, pour illustrer les performances et la robustesse des méthodes de recherche.

Mots clés: Réseau sur puce, trafic garanti, trafic au mieux, optimisation combinatoire, programme linéaire en nombres entiers, problème des K-plus courts chemins, reconfiguration dynamique, problème de flot maximum, graphe spatio-temporel, heuristique, recherche locale, algorithme évolutionnaire, algorithme mémétique

Abstract

We define and study a combinatorial optimization problem and mixed-integer linear programming, that models multi-path routing in a Network-on-Chip with guaranteed traffic. Based on time division multiplexing, the model allows to avoid collisions, deadlocks and livelocks in irregular network topologies, while minimizing latency. An extension of this multi-path problem is also presented that allows dynamic reconfigurable routing at execution time. In that case, independent sets of valid routes are pre-computed in such a way they can be interchanged during execution with no impact on the existing traffic, while reusing all the vacant and free time-slot resources.
A time-expanded graph approach is retained for the solution process. First, we present a set of basic operators to compute shortest paths. They can be a greedy parallel construction heuristic, neighborhood operators, and a modified Dijkstra algorithm in a time expanded graph that allows computing a single path in an occupied Noc in pseudo-polynomial time. Then, to solve all the problems, operators are introduced and combined within three iterated local search methods that can quickly generate feasible solutions, an evolutionary algorithm based on population conferring diversity solutions in search of solutions and a memetic algorithm, taking advantage of the benefits of the previous two.
Experiments are done on a set of benchmarks representative of real life applications, and instances of artificial applications randomly generated from real cases, to illustrate the performance and robustness of research methods.

keywords: Network on Chip, guaranteed traffic routing, best effort, combinatorial optimization, mixed-integer linear programming, K-Shortest Paths Problem, dynamic reconfiguration, maximum flow problem, time-expanded graph, heuristic, local search methods evolutionnary algorithm, memetic algorithm

n d'ordre : 273
Université de Bretagne-Sud
Laboratoire des sciences et technologies de l'information, de la communication et de la connaissance
Centre de recherche Christiaan Huygens BP 92116 F-56321 Lorient Cedex FRANCE
Tél: +33 (0)2 97 87 45 60
Fax: +33 (0)2 97 87 45 27

Remerciements

Je remercie M. Aziz Moukrim, Professeur des Universités, Université de Technologie de Compiègne, qui me fait l'honneur de présider ce jury.

Je remercie M. Lionel Amodéo, Professeur des Universités, Université de Technologie de Troyes et M. Philippe Lacomme, Maître de Conférences HDR, Université Blaise Pascal, Clermont-Ferrand, d'avoir bien voulu accepter la charge de rapporteur.

Je remercie M. Claude Yugma, Chargé de Recherche, École des Mines de St Étienne, d'avoir bien voulu accepter la charge d'examinateur.

Je remercie M. Jean-Philippe Diguet, Directeur de Recherche, CNRS, Lab-STICC, Université de Bretagne-Sud, Lorient, d'avoir bien voulu accepter la charge d'invité et examinateur.

Mes plus sincères remerciements à M. Marc Sevaux, Professeur des Universités, Lab-STICC, Université de Bretagne-Sud, Lorient, M. Oumarou Sié, Maître de Conférences CAMES, LTIC, Université de Ouagadougou, Burkina Faso, M. André Rossi, Maître de Conférences HDR, Lab-STICC, Université de Bretagne-Sud, Lorient et M. Jean-Charles Créput, Maître de Conférences HDR, Université de Technologie de Belfort-Montbéliard, qui ont dirigé ma thèse. Leur aide, conseil et guide m'ont été précieux pendant ces quatre années.

Ma reconnaissance à Éric, pour les contacts ayant permis d'être accueilli au Lab-STICC.
Ma pensée va à feu Seydou Zerbo, mon père, qui durant sa courte vie a su construire en moi une source d'énergie.
Je remercie ma mère, Boussitoa Fanta, qui a su me sortir des tempêtes de la vie au moment où j'étais le moins capable à les affronter.
Mes chaleureux remerciements à mon épouse, Aïchatou, à ma fille Fodè, dont le prénom évocateur de patience a caractérisé sa mère et elle, durant mes absences et parfois mon indisponibilité pendant mes présences.
Je remercie mes frères et sœurs Zerbo de leurs soutiens.
Je remercie mes amis et frères Seglaro Somé, qui a guidé mes premiers pas comme enseignant à l'université, et Ibrahiman Sakandé pour ses encouragements.
Je remercie, Matthieu et Clémence, qui m'ont ouvert les bras pendant mes séjours en France.
À tout mes ami(e)s, je dis merci pour leurs soutiens multiformes.

Sommaire

Introduction Générale

Contexte

La notion de réseau sur puce (NoC*) est maintenant une solution adoptée dans la technologie des semi-conducteurs dans laquelle le principe de réseau joue un rôle important [11, 3, 1]. Avec la réduction à l'échelle nanométrique de la taille des transistors, les concepteurs mettent au point des circuits intégrés complexes hétérogènes, incluant plusieurs éléments fonctionnels sur un même support de silicium, connus sous le nom de système sur puce (SoC*) ou système multiprocesseur sur puce (MPSoC*). Dans de tels systèmes, les solutions traditionnelles basées sur un bus partagé ont atteint leur limite évolutive et font place aux architectures de NoC comportant un réseau d'interconnexion avec des liens courts. De là découlent des problèmes de conception nouveaux. Comme dans les réseaux informatiques ou les réseaux de transport terrestre, la conception de systèmes efficaces de routage, de communication ou de transport, est une question cruciale pour permettre une garantie de la bande passante du trafic, éviter les collisions et les interblocages. Dans les NoC, les approches à garantie de trafic (GT*) et à trafic au mieux (BE*), sont souvent opposées [23]. Une caractéristique clé du routage GT est de régler le routage sans conflit au moment du calcul d'itinéraire, alors que le routage BE traite ces mêmes problèmes en cours de transport. Il est admis que les réseaux BE fournissent de bonnes performances moyennes, mais que les performances pour les pires cas sont très souvent difficiles à prévoir. De plus, pour éviter les possibilités d'interblocage, les réseaux de type BE impliquent des restrictions sur l'acheminement et/ou des coûts supplémentaires dus à l'éclatement des liens en canaux virtuels [6]. Au contraire, les méthodes GT assurent le respect des exigences temps réel de l'application, mais elles impliquent souvent un sur-dimensionnement de la bande passante du trafic entre les nœuds source et destination, et leur mise en œuvre souffre d'un manque d'adaptabilité dynamique aux fluctuations de trafic. Principalement, l'approche GT permet d'éviter la possibilité de conflits et d'interblocages par un calcul préalable des chemins et la synchronisation des échanges. C'est dans ce contexte des réseaux sur puce GT que se situent les travaux présentés dans ce mémoire. Plus précisément, nous nous plaçons dans le cadre de la méthodologie de conception de NoC μSpider II proposée par [9].

Objectifs et contributions

Dans le cadre de cette thèse, nous avons mené des travaux que l'on peut classer suivant quatre types d'objectifs. Leur atteinte peut être évaluée à travers les différentes contributions que nous présentons dans ce manuscrit.

Un premier ensemble d'objectifs est la formulation du problème de routage GT dans le NoC en tant que problème d'optimisation combinatoire précisément défini et

formalisé. La technique de routage utilisée est le multiplexage temporel dans lequel un ensemble de messages origine/destination constitués d'un nombre variable de paquets se partagent une plage de temps et utilisent des instants de passage disjoints au travers des liens du réseau. Le problème est cyclique dans la mesure où les messages sont émis régulièrement suivant un cycle de temps. Ce cycle de temps définit la plage de temps partagée entre les communications. Pour chaque nœud source, une table d'accès multiple à répartition dans le temps (TDMA*) indique les instants d'émission des messages vers leurs destinations respectives. Cette technique de multiplexage temporel dans un NoC a déjà été présentée et utilisée par exemple dans [47, 46, 18, 27, 44, 63]. L'apport que nous proposons par rapport à ces travaux en architecture de SoC, consiste à formuler précisément le problème de communication en le reliant au domaine de l'optimisation combinatoire et aux problèmes standards de routage de la littérature qui s'en rapprochent. Nous verrons par exemple que le problème peut être vu comme une combinaison d'un *Bin Packing* et d'un problème standard de K-plus-courts chemins avec contraintes. Nous montrons que le problème est NP-difficile au sens fort.

Un deuxième ensemble d'objectifs recherchés dans cette thèse consiste en l'extension du problème de routage afin de permettre la reconfiguration dynamique des chemins de communication lors de l'exécution de l'application sur le NoC. La différence par rapport au problème précédent, que nous appelons problème de routage standard, est que plusieurs tables TDMA spécifiant chacune une configuration de communication peuvent être associées à un même nœud source. Nous supposons que la phase d'optimisation a lieu au moment de la conception, mais que les changements de configuration d'émission peuvent se produire pendant la période d'exécution du NoC. Il s'agit donc de formaliser la définition de ce problème de routage avec reconfiguration dynamique et d'en étudier les propriétés. Une difficulté du problème est de permettre une réutilisation maximale des intervalles de temps libérés lors d'un changement de configuration d'émission, sans que cela entraine de conflit entre paquets. Nous proposons une solution à cette difficulté.

Un troisième ensemble d'objectifs est de tenter de caractériser les difficultés inhérentes à l'approche GT, ainsi que les potentialités et limites des mécanismes de reconfiguration dynamiques proposés. Nous procédons pour cela à une analyse de la tâche du concepteur lorsqu'il spécifie des flots de communication valides (messages et paquets), reconfigurables ou non, qui sont la donnée d'entrée des problèmes de routage GT. Nous verrons que la tâche n'est pas aussi aisée qu'il n'y paraît puisqu'elle s'apparente à la résolution (implicite) d'un problème de flot maximum non trivial. Nous en déduisons des limites du mécanisme de reconfiguration dynamique autorisé et proposons des pistes pour surmonter ces limites. Nous étudions dans le même temps les liens entre l'approche GT proposée et l'approche de type BE. Nous proposons de les considérer comme deux types d'approches complémentaires de routage pouvant éventuellement être combinées au sein d'un même NoC. De la spécification des flots de communica-

tion valides, nous déduisons également un algorithme de génération de jeux de trafic aléatoires qui seront utilisés pour les expérimentations et la validation des méthodes de résolution.

Un quatrième ensemble d'objectifs recherchés dans ces travaux réside dans la définition et la mise en œuvre de méthodes de résolution efficaces pour les problèmes d'optimisation combinatoires proposés. Ces approches sont la recherche d'une résolution exacte d'un programme linéaire en nombres en entiers du problème de routage standard, et des heuristiques et métaheuristiques pour les deux types de problèmes, le problème standard et son extension permettant la reconfiguration dynamique des chemins de communication.

Toutes ces méthodes utilisent un graphe spatio-temporel étendu (TEG*), qui contient une copie de l'ensemble des nœuds et des arcs du graphe initial pour chaque pas de temps discret considéré. Il permet de mémoriser l'état d'occupation du réseau aux différents intervalles de temps considérés. Le principe est déjà utilisé dans le routage multiple à répartition dans le temps [46]. Cette structure est également souvent rencontrée pour traiter des problèmes de plus courts chemins avec fenêtres de temps [34]. Elle permet de résoudre des problèmes de flots dans lesquels le temps joue un rôle important en appliquant des techniques algorithmiques standards développées pour des flots statiques. L'utilisation d'un TEG nous paraît indiquée dans le cas présent, dans la mesure où la taille mémoire nécessaire dans le cas des problèmes rencontrés dans la pratique reste raisonnable. A titre d'exemple, nous présentons un algorithme de type Dijkstra en mesure de calculer un plus court chemin unique dans un TEG en temps pseudo-polynomial. Nous verrons comment une telle procédure améliore considérablement les performances des algorithmes de résolution. De même, nous verrons comment une simple adaptation de la structure du TEG permet de traiter le problème avec reconfiguration dynamique des chemins.

En nous basant sur la structure du TEG, nous proposons des recherches locales et une approche évolutionnaire pour résoudre les deux problèmes de manière unifiée. Des opérateurs élémentaires de manipulation et construction de chemin sont tout d'abord présentés. Partant de cette base, trois versions de recherches locales sont proposées. Chacune correspond à un mode de parcours du voisinage différent. Ensuite, pour augmenter la diversité des solutions générées et surmonter les limites des recherches locales, un algorithme évolutionnaire est proposé faisant évoluer une population de solutions à l'aide des opérations de base et de sélections. Enfin, un algorithme de type mémétique est proposé qui consiste à combiner une des recherches locales précédentes au sein de l'algorithme évolutionnaire. Un ensemble d'expérimentations sur des cas réels d'application et des cas aléatoires a pour but de valider et évaluer les opérateurs et méthodes de résolution proposées. Un objectif supplémentaire implicite de cette évaluation est la définition d'un ensemble de jeux de tests standardisés et réutilisables

pour des évaluations ou comparaisons futures.

Organisation de la thèse

Globalement, les chapitres du document peuvent être regroupés selon trois parties distinctes. Une première partie composée des chapitres 1, 2, et 3 présente l'état de l'art du domaine et notre formalisation des problèmes de routage dans le NoC. Les chapitres 4, 5, et 6 constituent une deuxième partie exposant les méthodes, principes et algorithmes de résolution que nous proposons. Enfin, les chapitres 7, 8, 9 correspondent à une dernière partie du document dédiée aux évaluations et expérimentations de nos approches de résolution. Une conclusion générale termine le manuscrit.

Plus précisément, nous donnons un rapide descriptif des neufs chapitres.

Le chapitre 1 présente le concept de réseau sur puce, et la méthodologie du NoC μSpider II de [9] sur laquelle nous basons particulièrement nos travaux. De même, il offre un état de l'art des problèmes d'optimisation combinatoire rencontrés lors de la conception de ces systèmes.

Le chapitre 2 définit les notations qui seront utilisées dans cette étude. Nous présentons le problème de routage standard GT sous la forme d'un problème d'optimisation combinatoire que nous appelons « problème cyclique des K-plus-courts chemins sans conflits » et sa formulation sous la forme d'un programme linéaire en nombres entiers. Nous montrons que le problème est NP-difficile, et le rapportons à la littérature du routage dans les réseaux de communication et dans les réseaux de transport terrestre. Nous présentons le problème de la spécification des flots de communication origine/destination sous la forme d'un problème de flot maximum et terminons par un exposé des limites de l'approche.

Le chapitre 3 présente, sur la base des limites du routage GT standard, une formulation du problème autorisant la reconfiguration dynamique des chemins de communication. Nous le nommons « problème cyclique des K-plus-courts chemins reconfigurables sans conflits » . Nous présentons également le problème de flot maximum correspondant et présentons les limites et les extensions possibles de l'approche, dont le principe de la combinaison de trafic GT et BE.

Le chapitre 4 présente la solution de ces deux problèmes dans une approche unifiée se basant sur l'utilisation d'un graphe spatio-temporel étendu. Une condition nécessaire et suffisante est précisée permettant l'utilisation de plusieurs TDMA pour la reconfiguration dynamique du routage, tout en réutilisant les plages horaires sans conflit. Nous présentons également les lignes directrices du principe de fonctionnement des

recherches locales, de l'algorithme évolutionnaire et de l'algorithme mémétique proposés.

Le chapitre 5 présente les quatre opérateurs principaux formant l'ossature des méthodes de recherche que nous proposons, via leur pseudo-code. Ces opérateurs de base sont celui de la gestion des dates d'émission, celui de la construction parallèle gloutonne des chemins, celui de la procédure Dijkstra étendue dans le TEG et un opérateur de voisinage.

Le chapitre 6 présente les approches heuristiques et métaheuristiques proposées. Un premier algorithme de résolution est une recherche locale déclinée en trois versions suivant le mode d'examen du voisinage. Ces trois versions sont une recherche de type marche aléatoire, une recherche gloutonne et une recherche en profondeur. Un second algorithme est un algorithme à base de population de solutions incluant des opérateurs de sélection. Un troisième algorithme est une combinaison de la recherche locale avec l'algorithme évolutionnaire aboutissant à un algorithme mémétique.

Le chapitre 7 présente les jeux de tests structurés, le générateur de jeux de trafic aléatoires et la démarche d'évaluation expérimentale adoptée.

Le chapitre 8 présente une évaluation des recherches locales, à travers l'étude de l'impact des composants de base, leur comparaison avec la méthode exacte appliquée au programme linéaire en nombres entiers, l'évaluation sur des jeux de tests aléatoires et structurés, et l'application au cas du problème avec reconfiguration dynamique.

Le chapitre 9 présente une évaluation de l'algorithme évolutionnaire. Elle se fait par l'étude de l'impact de la procédure Dijsktra modifiée, l'application de l'algorithme évolutionnaire aux problèmes standard et reconfigurable et l'étude des versions min-sum et min-max du problème de routage standard.

Enfin, une conclusion générale termine le document et présente les perspectives de continuité des travaux présentés.

1

État de l'art

Ce chapitre présente le concept de réseau sur puce (NoC) et un état de l'art des problèmes d'optimisation combinatoire rencontrés lors de la conception de ces systèmes. L'évolution des technologies doit permettre de satisfaire les besoins croissants en puissance de calcul des applications des systèmes embarqués. Pour les satisfaire, les systèmes sur puce SoC, sur lesquels ces applications sont implantées subissent des évolutions, les rendant de plus en plus complexes. On est ainsi passé des systèmes mono-processeur à des systèmes multi-processeur implantés sur une même puce. Deux types d'architectures de communication sont utilisés pour véhiculer des données entre composants interconnectés. Ceux sont, les bus d'une part, et le réseau sur puce (NoC) d'autre part. Le concept de réseau sur puce (NoC) vise à éviter le goulet d'étranglement que constitue un bus lorsque le nombre de composants augmente. Il permet la connexion des composants selon des topologies variées définies par des routeurs reliés les uns aux autres et permettant le transfert des données par des liens courts.

Construire l'architecture de communication d'un NoC n'est pas une tâche triviale étant donné la multitude de paramètres qu'ils convient de déterminer simultanément afin de garantir une certaine qualité de service au meilleur coût. Une des clés de cette maîtrise des coûts est la segmentation du flot de conception du NoC en des étapes de conception indépendantes facilitant la modularité et la réutilisation. Chaque étape est un regroupement de paramètres cohérents et dépendants qui soulève un ensemble de problèmes à solutionner. Le modèle de référence (OSI*) fournit un standard de séparation de l'espace de conception d'un réseau sur puce (NoC) en sous-couches de services indépendants et flexibles et de niveaux d'abstraction croissants. L'élaboration progressive de l'architecture de communication de l'application en conformité avec le modèle OSI s'effectue alors en un certain nombre d'étapes séquentielles qui constituent le flot de conception. Les choix successifs de la conception vont déterminer des problèmes d'optimisation combinatoire variés avec différents niveaux de complexité. Nous prenons en exemple la démarche de conception proposée par [9] au sein de la plateforme de développent de réseau sur puce (NoC) μSpider II. De la spécification des communications à l'élaboration des chemins de routage, en passant par les étapes de construction et de dimensionnement, divers problèmes d'optimisation plus ou moins standards apparaissent, pour lesquels les méthodes de l'optimisation combinatoire et

de la recherche opérationnelle ont un rôle important à jouer. Les problèmes à résoudre sont des problèmes de placement, de dimensionnement, d'allocation de ressources, de flot, d'ordonnancement, et des problèmes de plus court chemin avec contraintes temporelles et de capacité pour ce qui nous concerne plus particulièrement.

Nous rappelons les composantes d'un SoC puis nous présentons les principales caractéristiques des NoC dans les sections 1.1 et 1.2 respectivement. Nous citons quelques exemples de NoC de la littérature et présentons les fondamentaux de leur conception. Nous présentons la méthodologie du NoC μSpider II de [9], sur laquelle nous basons nos travaux particulièrement, dans la section 1.3. Nous précisons ses caractéristiques et sa spécificité. Les cinq étapes du flot de conception de μSpider II sont détaillées dans la section 1.3.2. L'analyse du flot de conception nous conduit à identifier différents types de problèmes d'optimisation combinatoire que sa mise en œuvre soulève et à évaluer la nature de ces problèmes. Un exposé des principaux problèmes d'optimisation identifiés est réalisé dans la section 1.4. Cette liste n'est pas exhaustive étant donné les nombreuses combinaisons possibles et intriquées de problèmes que nous pouvons définir à partir d'eux. Nous terminons le chapitre par une conclusion.

1.1 Systèmes sur puces

Un système sur puce (SoC), multiprocesseur ou pas, est essentiellement composé d'une partie matérielle et d'une partie logicielle comme le montre la figure 1.1. D'une part, la partie matérielle est constituée par des processeurs standards, du plus simple au plus complexe (CPU*, DSP*, MCU*) qui traitent des données [30], et des blocs mémoires qui stockent des données en amont et en aval des processeurs. Ces différents blocs fonctionnels sont généralement appelés composants (IP*). D'autre part, la partie matérielle comporte le réseau de communication qui permet le transfert des données d'un bloc IP à un autre bloc IP. Les IP et le système de communication d'un système sur puce (SoC) interagissent par l'intermédiaire d'interfaces matérielles et d'adaptateur de protocoles [14]. La partie logicielle comporte des fonctions pratiques indispensables à l'application mais aussi dédiées à l'échafaudage matériel qui constituent le système d'exploitation (OS*) du système sur puce. L'OS est un isolant entre l'application et l'échafaudage matériel. Il comporte notamment les pilotes d'entrées/sorties. La dernière couche logicielle est l'application elle-même. Le système sur puce ainsi perçu permet d'entrevoir une élaboration découplée de la partie matérielle et de la partie logicielle, ainsi que de la partie communication [30].

Le vecteur de communication est l'élément qui nous intéresse ici. Les SoC utilisent principalement deux types d'architectures de communication pour véhiculer des données. Ce sont, les bus d'une part, et le réseau sur puce (NoC) d'autre part. Le bus est un câble métallique unique sur lequel les différents blocs se connectent. Il est donc partagé par les différents éléments et finit par constituer une entrave au développement de systèmes avec un nombre croissant de composants connectés. Pour pallier ses in-

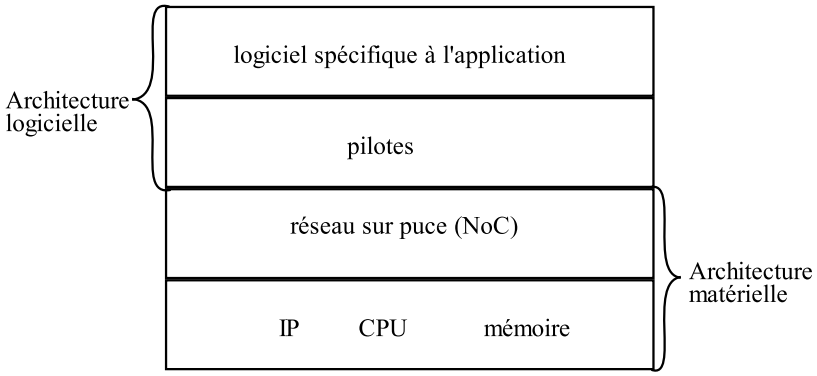


Figure 1.1: Architecture d'un SoC: partie matérielle et logicielle.

suffisances, différentes améliorations ont été proposées comme l'architecture à bus parallèle, dans laquelle chaque câble ou bus est dédié à un seul type de communication, ou l'architecture à bus segmenté composée de plusieurs bus mis en relation par des ponts [58, 14]. Cependant, le réseau sur puce(NoC) comme nouveau paradigme d'architecture de communication dans les systèmes sur puce (SoC) est apparu comme alternative au bus. C'est sur ce type de réseau de communication que nous focalisons notre attention dans les pages suivantes.

1.2 Réseaux sur puce

L'accroissement continuel du taux d'intégration des composants électroniques sur une même puce, autorisant la possibilité de connexion d'une multitude d'IP, a rendu moins opérant l'architecture à base de bus de communication. Le réseau sur puce (NoC) est alors apparu comme nouveau paradigme de communication. Nous en présentons quelques définitions rencontrées dans la littérature et décrivons ensuite leurs caractéristiques principales matérielles et logicielles afin d'y situer les choix de conception dont relève notre étude et justifier de leur intérêt. Nous verrons en quoi les choix dans les protocoles de routage sont dépendants des caractéristiques matérielles des routeurs et du type de topologie retenus. Nous verrons comment la conception vise à la recherche d'un compromis entre la surface du NoC, la consommation d'énergie et la qualité de service. Nous donnons des principes de base et renvoyons à [9, 61, 50, 28, 42, 12, 17] pour plus de compléments.

1.2.1 Définition d'un NoC

Diverses variantes de définition d'un NoC sont données dans la littérature. Par exemple, un NoC peut se définir comme un ensemble réparti entre des nœuds de routage

et des fils de communication [14]. Un NoC est un assemblage dans lequel des routeurs sont joints point à point par des fils dans le but d'orienter des données [24]. C'est un ensemble de nœuds routeurs diffusant des données entre eux à travers des voies de communication point à point nono ou bidirectionnelles [28]. Le NoC est la jonction d'un ensemble de liens permettant aux IP et aux routeurs d'échanger des informations entre eux [39]. Le NoC se définit selon [9] comme un système composé de plusieurs éléments de routage (routeurs) connectés selon une topologie spécifique. Le NoC s'appuie sur un envoi divisé des données à travers une toile de guides reliés entre eux par des liens [61]. Nous résumons toutes ces définitions par un schéma type d'architecture de NoC tel que donné dans la figure 1.2. Un Noc est un graphe orienté de communication connectant des routeurs, auxquels sont connectés des IP par le biais d'une interface réseau (NI*).

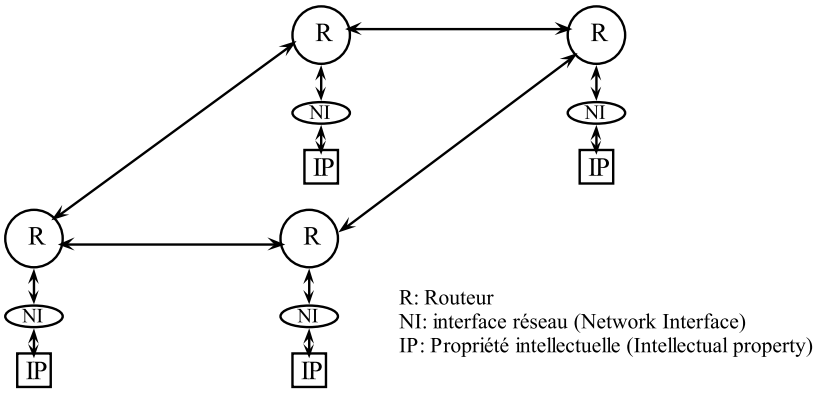


Figure 1.2: Schéma de principe d'un NoC.

1.2.2 Quelques NoC

Les réseaux sur puce sont présentés dans plusieurs études et états de l'art [17, 55, 12, 67, 28, 9, 14]. Différentes plateformes et méthodologies de conception et de réalisation sont présentées dans [9]. Parmi les principaux NoC rencontrés dans les domaines de la recherche et de l'industrie, nous pouvons citer:

- le réseau Æthereal développé par *Philips*;
- le réseau MANGO (*Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces*) de la *Technical University of Denmark*;
- le réseau Proteo qui est le fruit d'un consortium entre les universités finlandaises (*Tampere University of Technology et University of Turku*) et l'institut suédois (*Stockholm Royal Institute of Technology*) ;

- le réseau QNoC développé par le *Technion-Israel Institute of Technology*;
- le réseau SoCBus de l'université *Linköpings*;
- le réseau SPIN (*Scalable Programmable Integrated Network*) créé au sein de l'université Pierre et Marie Curie;
- le réseau STNoC ou Spidergon conçu par *STMicroelectronics*;
- le réseau XPIPES élaboré par l'université de Bologne;
- le réseau μSpider II élaboré par [9] à l'université de Bretagne Sud;

Table 1.1: Tableau de comparaison de quelques caractéristiques des NoC

Réseau	Topologie	Commutation	Algo. routage	Mémorisation	Qualité de service
XPIPES	Grille, Torus, Hypercube, Clos, Butterfly	Paquet	Déterministe	Wormhole	BE
SPIN	Arbre élargi	Paquet	Adaptatif	Wormhole	BE
Proteo	Anneau variable	Paquet	Déterministe	Wormhole	BE
STNoC	Spécifique	Paquet	Non spécifié	Wormhole	GT
SoCBus SoCBus	Grille 2D	Circuit	Distribué	Circuit Virtuel	GT
MANGO	Grille 2D	Circuit Paquet	déterministe	Circuit Virtuel	GT et BE
Æthereal	Grille 2D	Paquet	Déterministe	Wormhole(BE) Store-and-forward(GT et BE)	GT et BE
QNoC	Grille 2D	Paquet	Déterministe	Wormhole	4 qualités de service
μSpider II	Irrégulière	Paquet	Déterministe	Wormhole	GT

Par référence à la table 1.1, nous pouvons noter des paramètres qui caractérisent le fonctionnement des NoC et permettent d'en distinguer les variantes de réalisation. C'est ainsi que nous devons considérer le type de topologie, de nœud routeur, de liens, le type d'interface, la technique de routage, la politique de mémorisation et les techniques de commutation de paquet. Nous allons donner un aperçu de ces notions dans les pages qui suivent.

1.2.3 Architecture matérielle

Nous décrivons les principaux éléments qui déterminent l'architecture matérielle de communication d'un NoC. Ce sont la topologie, le routeur, le lien, et l'interface réseau qui relie un IP au NoC.

La topologie

L'architecture matérielle du NoC s'organise autour de sa topologie qui indique l'agencement des principaux composants matériels, leur disposition spatiale sur la puce et leurs interconnexions. Idéalement, la topologie est modélisée par un graphe orienté dont les sommets correspondent aux blocs IP et les arcs aux liens de communication

physiques entre les IP. Le choix de la topologie d'un NoC n'est jamais trivial. Il existe toujours une corrélation entre la topologie du NoC et les besoins en communications des composants, puisque la latence des communications peut dépendre de la distance physique entre les IP en communication sur la carte et des chemins de routage entre eux. Les NoC peuvent être classés suivant leur topologie. Les figures 1.3-1.4 présente des exemples de topologie adoptées dans les différents NoC. La topologie peut être régulière ce qui induit la possibilité de schémas de routage relativement simples, ou irrégulière ce qui autorise un meilleur dimensionnement du réseau pour l'application considérée mais requiert la mise en œuvre d'un routage approprié [6]. Cependant, de même que les chemins de routage dépendent des besoins en communication de l'application, de même la topologie en dépend.

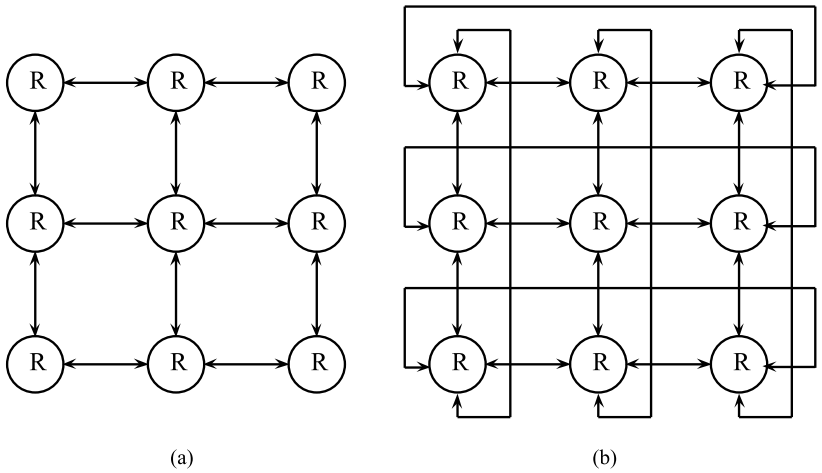


Figure 1.3: (a) Topologie 2D mesh. (b) Topologie 2D mesh torus.

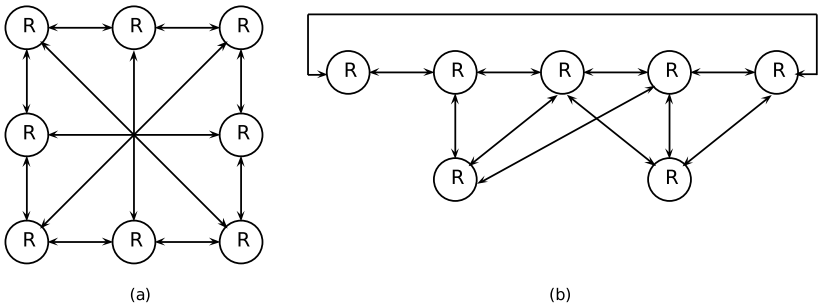


Figure 1.4: (a) Topologie Spidergon. (b) Topologie irrégulière.

Le routeur

Le routeur canalise ou oriente les données, du port d'entrée vers le bon port de sortie suivant le protocole sélectionné et la stratégie de routage choisie. C'est un composant physique de la puce dont la fonction se limite à retransmettre des données d'une direction d'entrée vers une direction de sortie. Il possède des ports d'entrée et de sortie qui sont mis en relation suivant le besoin de routage spécifié par le paquet de donnée à transférer. Il peut posséder des buffers de mémorisations de taille plus ou moins importante suivant le protocole ou le mécanisme de routage choisi. Les capacités de transfert et de mémorisation s'expriment en largeur, suivant le nombre de bits simultanément transmis via les liens physiques, et en profondeur, suivant le nombre de mots (ensembles de bits) mémorisés dans le routeur. Un routeur est connecté à d'autres routeurs ou bien à une IP via une interface standardisée (NI).

Le lien

C'est un lien physique qui permet le passage des données d'un routeur à l'autre ou vers un IP. On parle également de canal de communication. Le nombre de fils physiques utilisés détermine la bande passante maximum par cycle d'horloge. Il peut se présenter en mono ou bi-direction et permettre l'implantation d'un ou plusieurs canaux virtuels.

L'interface réseau

La communication directe n'est pas souhaitable entre les IP et les nœuds routeurs du NoC à cause de l'hétérogénéité des protocoles. Aussi, l'interface réseau NI, convertit le langage des IP en langage compréhensible par le NoC et vice versa. C'est un découpleur des traitements liés aux IP et aux communications à réaliser dans le réseau. La NI fournit un standard d'utilisation du NoC qui permet la modularité en dissimulant les détails d'implantation du NoC. C'est aussi un réducteur des messages en paquets pour l'émission et un assembleur des paquets en messages pour la réception.

1.2.4 Architectures et protocoles de routage

La partie communication du NoC réside dans un ensemble de protocoles implantés dans les routeurs et les NI. Ils régissent le transfert des messages d'un point à un autre du réseau. Les protocoles incluent le routage proprement dit, la commutation et la politique de mémorisation.

Les messages et paquets

Du point de vue de son routage et acheminement vers une destination, un message émis par un IP est généralement scindé en parties plus petites suivant le schéma message-paquet-*flit-phit* illustré à la Figure 1.5. Le paquet résulte du découpage d'un message en parties indépendantes émises séquentiellement. Un paquet est lui-même fractionné

en *flits* (*Flow Control Unit*) servant d'unité de contrôle de flux. Le paquet comporte un *flit* d'entête et un corps de taille variable, composé d'une suite de *flits* contenant les charges utiles du message. L'entête renseigne sur le cheminement requis et la destination du paquet. Les *flits* peuvent à leur tour être subdivisés en *phit* (*Physical Unit*) qui est l'unité de base pouvant être autorisée à passer par un lien physique du réseau en un cycle de temps. Le contrôle de transmission se fait à la fréquence des *flit*, alors que les données circulent à la fréquence des *phit*. Le paquet est l'unité de transmission indépendante dont les *flits* qui le composent doivent en principe rester contiguës dans le réseau et transmis en séquence. Pour notre part, dans le corps de cette thèse, nous ne considérons que deux niveaux de segmentation qui correspondent à la dichotomie paquet-*flit* usuelle [9] pour lesquels nous utilisons les termes message-paquet par convention de langage ainsi qu'illustré à la Figure 1.6.

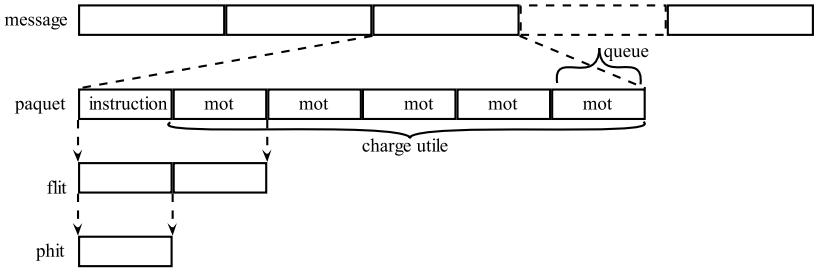


Figure 1.5: Segmentation message/paquet/flit/phit.

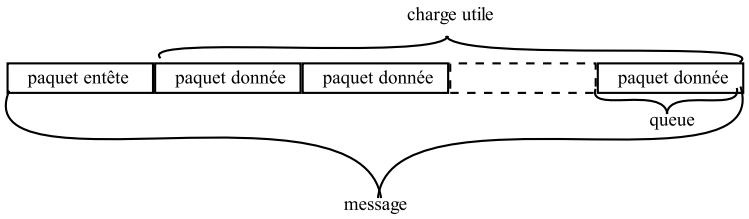


Figure 1.6: Segmentation message/paquet.

La commutation

La commutation définit comment sont attribuées des ressources de communication sur un chemin qui sera emprunté par un message de sa source à sa destination. Il en existe principalement deux modes que sont la commutation de circuit et la commutation de paquet.

- **Commutation de circuit**
  Toutes les ressources de communication sur le chemin emprunté par un message,

lors de son transfert de sa source à sa destination, lui sont exclusivement réservées durant tout son transfert. Aucun autre message ne peut accéder à une partie de ces ressources si ce n'est après leur dés-allocation suite à la transmission complète du message responsable de la réservation [50];

- **Commutation de paquet**
  Dans ce mode de commutation aucune réservation de ressources de communication n'est effectuée par le message. L'allocation de ressources est faite aux paquets du message au fur et à mesure de leur progression en fonction de la disponibilité de celles-ci. Le message sera donc préalablement mis en paquets avant son injection dans le réseau [61].

La fonction de routage

C'est l'ensemble des mécanismes par lesquels le chemin ou les chemins qui seront empruntés par les paquets d'un message, entre la source et la destination dudit message, sont déterminés [61]. La fonction de routage peut être distribuée dans les routeurs ou bien dépendre de la source du message. Dans le cas du routage distribué, la destination en sortie d'un routeur dépend uniquement du routeur courant et de la destination du paquet. Cette technique entraîne la mémorisation des tables de routage dans les routeurs eux-mêmes. L'algorithme de routage peut être, soit déterministe, lorsque le choix du chemin est unique, soit adaptatif, lorsque le choix du routage dépend de l'état courant des canaux de sortie du routeur. Dans le cas du routage à la source, le chemin est calculé par la source et le paquet d'entête contient une spécification de ce chemin. Un routeur se contente alors de transférer le paquet courant vers la sortie spécifiée par le chemin. Cette technique permet d'éviter la constitution de tables de routage dans les routeurs et apporte plus de souplesse dans le choix des possibilités de cheminement pour une même destination.

La politique de mémorisation

De par son principe de réservation exclusive de ressources de communication pour un message, la commutation de circuit permet d'éviter les conflits dans les liens du réseau lors des communications. Par contre, dans la commutation de paquets où l'accès aux ressources de communication par les paquets se fait sans réservation initiale, des conflits peuvent exister. Pour les réduire et éventuellement les éviter, on met en place des buffers de mémorisation des paquets de message. Cette mémorisation permet l'attente d'une libération de ressource pour le passage du paquet. Il existe plusieurs types de contrôle de flux de données qui déterminent la quantité de mémoire requise dans les routeurs. On peut en retenir principalement trois qui sont:

- ***Store-and-Forward* (SAF)**: chaque routeur du réseau attend la réception de la totalité du message, le stocke entièrement avant de le transmettre au suivant;

- ***Virtual Cut Through* (VCT)**: cette technique a été introduite pour réduire la latence du SAF. Un composant réseau fait suivre les paquets d'un message au plus vite, mais il peut stocker tout le message s'il y a une contention;
- ***Wormhole***: cette politique consiste à acheminer le message décomposé en paquets, sans possibilité de stocker la totalité du message dans un routeur. Les paquets sont transmis de l'entrée vers la sortie des routeurs de manière pipeline. Seul le paquet de tête contient l'information de routage, aussi les différents paquets composant le message doivent rester contiguës dans le réseau. Ils sont mémorisés et répartis dans les canaux le long du chemin suivi. Cette technique de routage entraîne des risques supplémentaires de contention mais réduit la mémoire nécessaire dans les routeurs. Étant donné ses avantages, cette méthode est la plus utilisée dans les réseaux sur puce (NoC) et il s'agit de celle retenue dans le cadre du NoC μSpider II et de nos travaux.

1.3 Le NoC micro-Spider II

1.3.1 Présentation

Le NoC μSpider II est à la fois une architecture et une méthodologie de conception de réseau sur puce (NoC) qui permet l'utilisation de topologies irrégulières adaptées aux spécifications de l'application et met l'accent sur la possibilité de communication avec une qualité de service à garantie de trafic (GT). C'est un réseau à commutation de paquet, dont le routage est déterminé à la source, avec une transmission de type *wormhole*. Il garantit des performances prédéfinies en bande passante en utilisant le principe d'accès multiple à répartition dans le temps par l'utilisation de tables TDMA. Ce principe permet de pré-ordonnancer l'envoi de paquets et de réserver des instants d'occupation (*time-slots*) des liens lors du calcul de l'ensemble des chemins. La technique vise à garantir une bande passante donnée tout en permettant l'absence de tout conflit entre paquet. En outre, le NoC μSpider II prévoit des mécanismes de reconfiguration dynamique des tables d'émission TDMA. C'est en lien avec ces mécanismes que s'articulent nos travaux dans cette thèse. Ce type d'architecture vise à garantir des performances tout en réduisant au maximum la complexité des routeurs. La contrepartie de cet allègement des routeurs se retrouve néanmoins dans la nécessité d'une optimisation préalable des chemins afin d'éviter les conflits, ce qui implique de résoudre un problème d'optimisation non trivial. Ce sont ces procédures et problèmes d'optimisation que nous traitons dans cette thèse. Les spécificités du Noc μSpider II peuvent être résumées par les points suivants :

- **Topologie irrégulière** Le NoC μSpider II permet une topologie irrégulière ce qui peut apporter un gain en surface et consommation non négligeable par rapport à une topologie régulière [6]. C'est l'application et un certain nombre de ses paramètres qui guident sa construction.

- **Interface réseau adaptative** La connexion entre un IP et le NoC se fait par l'intermédiaire d'une interface réseau NI. Celle-ci effectue la préparation des messages à transmettre et leur mise en paquets. Elle comporte des mécanismes de reconfiguration des tailles de buffers, et des choix des tables TDMA actives.

- **Routage à la source et multiplexage temporel** Le chemin de données est mémorisé par la NI émettrice et est codé dans le paquet d'entête. Lorsque le multiplexage temporel est réalisé et conduit à l'absence de conflits, aucun buffer mémoire n'est requis dans les routeurs. Une bande passante est alors garantie pour l'ensemble des communications.

- **Contrôle de flux à effet domino** Bien que le routage puisse être en principe déterminé sans conflit par la technique du multiplexage temporel, les concepteurs ont néanmoins prévu la possibilité de relâcher cette contrainte et de permettre la gestion des conflits par le contrôle de flux à effet dominos. Lorsqu'un routeur ou une NI n'est pas en mesure de consommer ou de stocker des données provenant d'un routeur en amont sur un de ces ports d'entrée, il émet un signal de blocage spécifique vers le routeur amont. À son tour celui-ci mémorise le paquet en cours de transmission dans un buffer local et relaie le signal de blocage vers son prédécesseur sur le chemin de donnée courant. Ainsi, le signal de blocage se propage pas à pas et dans le sens inverse du chemin de la transmission, d'où le nom d'effet domino donné au contrôle de flux de bout en bout du NoC μSpider II. Le contrôle de flux permet une communication de type trafic au mieux (BE) dans laquelle les conflits se présentent au hasard. Pour rendre le réseau trafic garanti (GT) compatible avec le trafic au mieux (BE), il suffit alors de s'assurer de l'absence d'interblocage de la fonction de routage.

1.3.2 Flot de conception du NoC micro-Spider II

Le flot de conception est l'ensemble des étapes à suivre pour la construction d'un NoC. C'est un regroupement de tous les processus pouvant intervenir dans la conception du NoC en sous ensembles renfermant des problèmes cohérents et indépendants. La résolution des problèmes à chaque étape conduit de la conception à l'implantation du NoC, par exemple sur (FPGA*). La définition d'un flot de conception permet de maîtriser la complexité de l'espace de construction du NoC en vue de satisfaire les critères de performance requise en termes de débit, de latence, d'énergie consommée et de surface occupée [54].

La génération du flot de conception du NoC μSpider II part de l'idée d'une architecture multiprocesseur adaptée au type de l'application et au partitionnement de ses tâches sur les différentes unités de traitement. Par conséquent, la conception du NoC commence par une analyse de l'application et une spécification du graphe de communication. Les étapes de conception du NoC μSpider II sont la spécification, la construction, le dimensionnement des TDMA et des mémoires, et l'allocation des chemins

spatio-temporels [9].

Spécification

L'objectif ici est de donner les moyens au concepteur de déterminer la fréquence du réseau, le nombre d'IP connectés au réseau, le partitionnement des tâches de l'application sur les IP, ainsi que l'ordonnancement des communications attribuées à chaque tâche. Cette détermination est effectuée à l'aide de graphes de dépendances (CDG*). Chaque graphe caractérise une tâche avec des sommets représentant soit un traitement spécifié par son temps d'exécution, soit une communication. Une communication est un flux typé d'information quantifiée en nombre de paquets entre deux tâches différentes, une émettrice et une réceptrice. Les arcs entre les différents sommets représentent un ordre d'exécution séquentiel ou parallèle de ceux-ci. La tâche modélisée sous la forme d'un graphe de dépendance comporte la taille des paquets émis et reçus, son temps d'exécution et l'identifiant de l'IP qui l'exécute. Résumer l'application à l'aide de ces graphes de dépendance permet la mise en évidence du parallélisme d'exécution interne à la tâche et de regrouper ses besoins en bande passante dans des spécifications de flots de communication. Ces spécifications de messages avec leur quantité de paquets peuvent être résumées sous forme d'un graphe orienté pondéré que nous nommons Graphe de Communication de l'Application (ACG*). Les sommets du graphe sont des IP, tandis que les arcs pondérés sont des bandes passantes requises entre IP. Ce type de graphe de communication constitue pour une part la donnée d'entrée des problèmes d'optimisation de routage étudiés dans cette thèse. Nous verrons notamment que la construction même d'un ACG compatible avec le principe de multiplexage temporel n'est pas toujours une tâche aisée et revient à résoudre un problème de flot maximum.

Construction

L'étape de construction s'appuie sur la bande passante maximale et réelle des différentes communications obtenue via les graphes CDG. A partir de ces données, le concepteur construit progressivement la topologie du réseau en appliquant l'un des algorithmes de conception, selon la nature de l'application et le type d'adressage des mémoires dans l'architecture multi-processeur. Les composants devront être d'autant plus rapprochés sur la carte que la fréquence et le volume de leurs échanges est important. Le concepteur ajoute des routeurs et des liens de manière à éviter un trop grand nombre d'entrée/sortie sur un même routeur. Il peut également s'appuyer sur les résultats des algorithmes de routage (dernière étape) pour ajouter ou supprimer des routeurs et des liens physiques.

Dimensionnement de la table TDMA

L'étape de dimensionnement de la table TDMA consiste à calculer le nombre d'intervalles de temps à réserver pour chaque communication selon sa bande passante et pour

chaque interface réseau. Le rôle principal de l'interface réseau est d'ordonnancer l'envoi des données vers le réseau afin de garantir une qualité de service conforme aux contraintes de bande passante. Cet ordonnancement est mis en œuvre avec l'intégration du mécanisme d'accès multiple à répartition dans le temps (table TDMA), dont le principe est de découper le temps disponible entre les différentes communications de la NI. Ensuite, intervient une phase d'uniformisation de la taille des TDMA de toutes les NI. Elle est faite de telle sorte que la taille des TDMA définisse un cycle d'émission dont la longueur en nombre de slots est partagée par toutes les NI. Chaque NI émet alors des paquets aux instants spécifiés par la table TDMA et cela de manière cyclique selon la taille (commune) de la TDMA.

Dimensionnement des mémoires

L'étape de dimensionnement des mémoires détermine les tailles des mémoires tampons des interfaces réseau, et les tailles des mémoires caches qui participent à la liaison entre l'IP et le réseau.

Allocation des chemins

Il s'agit de calculer les dates d'émission relatives des messages dans chaque table TDMA et les chemins de routage associés selon le principe du multiplexage temporel, en s'appuyant sur la topologie du NoC, la taille de la table TDMA et les spécifications de messages avec leur quantité. La possibilité de reconfiguration dynamique des tables TDMA est une option supplémentaire offerte par le NoC μSpider II. La proposition d'algorithmes heuristiques et métaheuristiques d'optimisation pour résoudre ces problèmes fait l'objet de notre travail.

1.4 Problèmes d'optimisation combinatoire dans le NoC

Le résultat recherché lors de la conception d'un NoC est l'obtention d'un compromis satisfaisant entre des critères de performance en termes de bande passante, de latence, d'énergie consommée et de surface occupée [54]. Il s'agit d'ajuster la taille du réseau et des ressources physiques nécessaires au bon fonctionnement du réseau de communication. Ainsi que détaillé dans [12, 9], le dimensionnement du NoC consiste à déterminer la taille et le nombre de ressources du réseau de communication permettant de satisfaire une certaine qualité de service en adéquation avec les besoins de l'application. Parmi ces ressources figurent les tailles de buffers d'entrée/sortie, le nombre de routeurs et de liens physiques, la taille des tables TDMA pour du trafic garanti. Les nombreux paramètres à ajuster font de la problématique de dimensionnement un problème multi-critère complexe impliquant directement ou indirectement la prise en compte du routage et du placement des IP sur le réseau pour aboutir à une structure optimale du réseau. Les problèmes rencontrés sont généralement intriqués dans la mesure ou

des choix à une étape de la conception ont une influence aux étapes suivantes. Des mauvais choix au niveau de la topologie auront par exemple une incidence au niveau du routage. Cependant, en pratique l'étape de routage dépend d'une topologie donnée qui peut être affinée de manière interactive. Nous présentons ci-après quelques uns des problèmes d'optimisation combinatoire considérés comme les plus importants en lien avec la conception d'un NoC et qui sont en grande majorité des problèmes de placement et de routage, ou encore des problèmes de détection d'interblocage.

1.4.1 Problèmes de placement

Le problème de placement, ou *facility location problem*, est sous-jacent à la détermination de la topologie. La topologie est induite par la disposition spatiale des composants connectés pour en faire le réseau sur puce (NoC). Si le choix de la topologie initiale est effectué aléatoirement, la détermination de la position finale des IP revient à un problème de placement. Souvent le problème de placement est présenté comme une variante du problème d'assignation quadratique (QAP*) [2]. Dans ce cas, le routage est traité de manière implicite, il s'agit de minimiser les distances entre IP pondérées par le besoin en bande passante de communication. Ainsi, plus le volume d'échanges entre deux IP est important, plus la distance physique entre les deux IP doit être courte. Généralement, le problème de placement est traité par des heuristiques ad hoc [47, 46, 18, 27, 44, 63] car il s'agit d'un problème NP-difficile. En particulier le QAP est considéré comme un des problèmes parmi les plus difficiles à résoudre en pratique. Seules des instances de très petites tailles sont résolues de manière exacte. Le principe d'une corrélation entre distance physique et besoin en bande passante est pris en compte dans [9] dans la phase de construction de la topologie et du choix des routeurs et des liens à ajouter.

Le placement et le routage peuvent être considérés de façon plus ou moins simultanée lors de la conception du NoC. C'est le cas dans [51, 31] où le choix de l'architecture du NoC se limite à un problème de placement validé par l'exploration de différents algorithmes de routage. Dans [52, 38, 16], le placement est réalisé sur plusieurs topologies pour en faire le choix de la meilleure par comparaison des performances obtenues. Plusieurs paramètres influent sur l'efficacité de cette méthode tels que les critères de comparaison, le nombre de topologies comparées, l'algorithme de routage utilisé. Une autre façon de traiter le problème de placement est le *floorplanning* [36]. C'est un problème de placement sans superposition des IP dont l'objectif est de minimiser la longueur totale des connexions entre les IP.

Dans [12], le placement et le routage sont menés ensemble comme technique d'adéquation algorithme architecture. Le problème revient à apparier deux graphes dont l'un pour l'application et l'autre pour l'architecture. L'adéquation est définie par des critères de respect de bande passante pour les chemins de données calculés. Généralement, les chemins de routage sont explicitement calculés selon une stratégie de routage relativement simple telle que le routage XY pour garantir l'absence de *dead-*

lock dans le cas du trafic BE. Le placement est valide lorsque les contraintes de bandes passantes sont respectées. Ce type d'approche placement/routage est également considérée dans [27] et combinée au routage à multiplexage temporel de type GT, ce qui conduit à un espace de recherche encore plus contraint. Les auteurs abordent le problème par l'insertion séquentielle et progressive des flots de communication, sans retour arrière. La méthode heuristique qui en découle peut donc s' apparenter à une démarche gloutonne. On notera que les versions du problème de placement sont diverses et variées suivant la manière dont elles intègrent la prise en compte du routage. Mais étant donné la taille de l'espace de conception à explorer, la plupart des méthodes se ramènent à des heuristiques qui vont restreindre et simplifier les choix possibles de conception à chaque niveau. Pour notre part, le cadre est celui où le placement préalable des IP est déjà effectué. Nous mettons donc l'accent sur une optimisation poussée du routage et considérons sa possibilité de reconfiguration dynamique.

1.4.2 Problèmes de routage

Le routage est un service de transmission de données offert par le NoC. Ses propriétés ont un impact crucial pour la qualité de service du NoC. Il existe principalement deux types de qualité de service (QoS*) recherchées. Il s'agit du trafic garanti (GT) et du trafic au mieux (BE). La distinction entre ces deux types de qualité de service tient principalement en ce que l'approche GT se base sur une allocation de ressources suffisante pour les scénarii de trafic au pire cas, tandis que l'approche BE prend en considération seulement la moyenne de tous les scénarii de trafic. Dans tous les cas de figure, il est présupposé que tout transfert de données se fait sans dégradation et sans déperdition et que l'arrivée des messages et paquets à destination se fait au bout d'un temps fini dans l'ordre de leur émission. Nous décrivons plus en détail ces deux types de qualité de service, leur mise en œuvre et les problèmes d'optimisation combinatoire qui en découlent.

Routage *Guaranteed Traffic*

L'approche GT est basée sur une réservation des ressources de communication de manière à garantir la bande passante et la latence quelles que soient les fluctuations en trafic. Un niveau de bande passante maximum supérieur aux besoins moyens de l'application est alloué pour chaque flux de communication entre IP. Il doit être suffisant pour tous les besoins temps-réels de l'application et ne peut être dépassé. L'approche GT est généralement mise en œuvre par la technique du multiplexage temporel et l'utilisation de tables TDMA [47, 46, 18, 27, 44, 63]. , ou encore par multiplexage spatial [43]. La technique du multiplexage temporel est la plus utilisée et celle que nous considérons dans cette thèse. Les messages sont émis par chaque IP selon un cycle de temps de longueur T commun aux IP, et à des instants successifs entre 0 et $T-1$ de telle sorte que les paquets chemineront dans le réseau sans aucun conflit. On dit que le trafic est sans contention ou sans conflit. Les problèmes d'optimisation combinatoires

que soulève cette approche combinent des caractéristiques de calcul de plus courts chemins multiples, de respect des capacités en bande passante, et d'ordonnancement des émissions.

Routage *Best Effort*

A l'inverse, l'approche BE repose sur une distribution des ressources de communication [67] sachant qu'aucune limitation de bande passante n'est effectivement imposée aux différents flux. Cela n'interdit pas néanmoins une allocation de ressources pour chaque chemin de communication en fonction du niveau de trafic moyen estimé requis. Le dimensionnement des ressources est effectué sur la base de la moyenne des contraintes de toutes les situations [12]. En revanche, les émetteurs sont autorisés à injecter des paquets de données même si cela risque d?entraîner un dépassement de la capacité du réseau sur des canaux en aval. Les conflits de paquets qui en résultent sont alors gérés par les routeurs selon une priorité tournante. Pour le calcul des plus courts chemins, seul le respect d'une bande passante moyenne est considéré, c'est au réseau de gérer les conflits ensuite en temps réel. L'ordonnancement préalable des dates d'émission de message n'a pas lieu d'être car aucune hypothèse de synchronisation n'est requise. Cependant, la problématique importante à traiter qui découle de la possibilité de conflits entre paquets est de garantir l'absence d'interblocage (*deadlock)* dans le réseau. Un *deadlock* dans le réseau de communication est une situation dans laquelle plusieurs messages ne peuvent plus avancer vers leur destination à cause de leur attente mutuelle des ressources qu'ils occupent eux-mêmes.

Que l'on recherche une qualité de service de type BE ou GT, divers problèmes d'optimisation combinatoire apparaissent. Ces problèmes peuvent être communs aux deux approches ou bien leur être spécifiques. Indépendamment de la topologie du réseau, le concepteur fait face à un problème de calcul de flot maximum lorsqu'il attribue des bandes passantes compatibles avec la capacité d'injection ou d'absorption de paquets dans le réseau. Le problème de flot maximum est un problème standard qui peut être résolu en temps polynomial. Nous reviendrons sur cette problématique de spécification du graphe de communication de l'application au cours de ce document. Une fois les bandes passantes spécifiées pour chaque communication point à point, il convient de déterminer les chemins de données pour le routage en étroite adéquation avec la topologie du réseau et sachant que plusieurs communications peuvent partager des liens en commun. Pour assurer un transfert des données fluide, le calcul des chemins doit tenir compte de la capacité en bande passante des liens physiques du réseau. Le non respect de cette capacité entraînerait une contention structurelle et un encombrement continu du réseau.

Les problèmes d'optimisation combinatoire les plus connus liés aux calculs des chemins origine/destination sont des problèmes de K-plus-courts-chemins avec contraintes [33], ou de flots multiples insécables (UFP*) [35]. Ces problèmes sont NP-difficiles. Dans le cas de trafic BE, résoudre un UFP peut-être suffisant pour en pratique

garantir le besoin moyen en bande passante, sachant que la contention et les conflits entre paquets présenteront un caractère occasionnel. Dans le cas du trafic GT, et de la technique de multiplexage temporel, tels que nous l'envisageons, une extension du problème UFP doit être considérée qui consiste en l'ordonnancement cyclique des envois de messages selon des dates d'émission préalables afin de permettre la synchronisation des transferts et l'absence totale de conflit. C'est ce problème étendu que nous étudions dans cette thèse. Il est présenté sous diverses formes et traité par des méthodes heuristiques ad hoc [47, 46, 18, 27, 44, 63]. Dans cette thèse, nous voulons notamment en donner une version standardisée et formelle avec des benchmarks types pour l'évaluation des heuristiques. Ce type de problème a également des analogies avec certains problèmes de routage avec fenêtres de temps dans les transports terrestres comme le *one-to-one shortest path problem with time windows* (SPPTW) [13], ou encore *les Automated Guided Vehicles* (AGV) [48].

1.4.3 Problème de détection de *deadlock*

Dans le cas du trafic BE en particulier, qui autorise la possibilité de conflits entre paquets, la problématique supplémentaire cruciale à traiter est celle de la prévention des interblocages. Cette prévention peut résulter de l'adoption d'un routage approprié comme le routage XY dans un réseau régulier, ou être obtenue par détection des cycles du graphe de dépendance [11] et leur élimination par l'usage de canaux virtuels. Notamment dans le cas de topologies irrégulières, la détermination du nombre minimum d'arcs communs aux différents cycles du graphe de dépendance et qui doivent être éliminés constitue un problème NP-complet appelé *minimum weight feedback edge set problem* [21]. Une fois les arcs identifiés, il convient de les éliminer du réseau tout en s'assurant du maintien de la connectivité du réseau [6, 65]. Des canaux virtuels peuvent être créés. Si le trafic BE permet une souplesse dans l'utilisation des ressources, sa mise en œuvre nécessite l'utilisation de buffers de mémorisation dans les entrées de routeurs et l'élimination des cycles du graphe de dépendance. Le trafic GT permet en revanche l'utilisation de routeurs simplifiés et sans mémorisation. Un enjeu de cette thèse est de tenter d'introduire plus de souplesse dans les techniques GT par la reconfiguration dynamique des chemins de communication non-conflictuels.

1.5 Conclusion

Nous avons présenté le concept de réseau sur puce (NoC) et les principales méthodes utilisées pour les concevoir. Le principe de leur architecture matérielle réside dans l'utilisation de routeurs interconnectés permettant d'assurer une communication fiable entre de nombreux composants IP. Les IP mettent en œuvre les fonctions de l'application et sont reliés au NoC via des interfaces standards NI. Le réseau sur puce (NoC) fournit des services de communication qui visent à répondre aux difficultés rencontrées par l'utilisation des bus de données. Contrairement au bus, l'architecture de

communication d'un NoC comporte des liens de communication courts entre des IP agencés et placés sur la puce selon une topologie adaptée à l'application et répartie sur la surface de la puce de manière régulière ou irrégulière. Nous avons présenté les principales techniques de commutation de paquet utilisées et les techniques de routage, en précisant lesquelles sont en lien avec nos travaux et leurs caractéristiques. Nous avons choisi comme cadre la méthodologie de conception du NoC μSpider II, la commutation de paquet de type *wormhole* et le routage *Guaranteed Trafic* (GT) par multiplexage temporel.

En présentant les spécificités du routage *Guaranteed Trafic* (GT) par rapport au routage *Best Effort* (BE) plus courant, nous avons mis en évidence un certain nombre de problèmes d'optimisation combinatoire rencontrés lors de la conception des NoC. Ces problèmes sont principalement des problèmes de placement (*facility location problem*), de calcul de plus courts chemins avec contraintes de capacité, d'ordonnancement et de synchronisation des échanges, et plus particulièrement pour le trafic BE des problèmes de détection et de réduction d'interblocage. Que ce soit pour le trafic BE ou le trafic GT, le respect des capacités en bande passante des liens de communication semble essentiel pour éviter toute possibilité de contention structurelle et durable. Alors que le réseau BE permet l'injection des paquets sans avoir à se préoccuper de la charge courante du réseau, et permet la présence de conflits entre paquets, en principe occasionnels, le réseau GT fixe une bande passante maximum spécifique pour chaque communication. Le respect des contraintes temps-réel de l'application entraîne alors un sur-dimensionnement du réseau basé sur des besoins au pire cas. Un des objectifs de cette thèse est d'apporter plus de souplesse dans l'utilisation d'un réseau GT par la reconfiguration dynamique des chemins de communications et des bandes passantes, et ainsi de permettre un meilleur ajustement des ressources tout en maintenant une qualité de service garantie avec un trafic sans conflit ni interblocage au sein d'une topologie irrégulière.

2

Problème de routage avec garantie de trafic

Le réseau sur puce (NoC) est une approche émergente de la technologie des systèmes multiprocesseurs sur puce MPSoC dans laquelle la recherche de chemins de routage efficaces est un des challenges parmi les plus importants [11, 3, 1]. Dans de tels systèmes, les solutions traditionnelles avec les bus partagés sont remplacées par des interconnexions avec des liens courts. Comme dans les réseaux d'ordinateurs ou dans les réseaux terrestres de communication ou de transport, l'issue critique est la prévision d'une bande passante à même de garantir la qualité du trafic, l'évitement des collisions, les interblocages (*deadlock*). Le routage de type Garantie de Trafic (GT) est une réponse à cette préoccupation. Le GT est la spécification d'une QoS dans le NoC qui doit se traduire par le respect des contraintes temps réels des applications, et donc d'une garantie de débit et de latence appropriés. La technique du multiplexage temporel, par l'usage de tables d'émission TDMA, est celle retenue ici pour mettre en œuvre cette QoS. Elle consiste à allouer à chaque communication des instants (*time-slots*) d'émission et de passage autorisés des paquets au travers des liens du réseau. Étant donné les contraintes multiples à satisfaire simultanément pour toutes les communications, la construction des chemins de routage n'est pas une tâche aisée.

Dans ce chapitre, nous modélisons le problème de routage (GT) dans le NoC sous la forme d'un problème d'optimisation combinatoire dont nous montrons qu'il est NP-difficile au sens fort. Nous l'appelons, « problème cyclique des K-plus-courts chemins sans conflits » (CKPP*). Nous donnons une formulation du problème en termes mathématiques usuels en théorie des graphes et nous le présentons également sous la forme d'un programme linéaire en nombres entiers. Nous analysons ensuite les contraintes que doivent respecter les spécifications de flot de communication pour constituer des données d'entrée valides du problème. En effet, indépendamment de la structure du réseau, les quantités de paquets spécifiées doivent respecter à la fois la capacité des IP émetteurs et la capacité des IP récepteurs. Ainsi, la spécification même de la donnée d'entrée du CKPP peut être représentée par la résolution d'un problème de flot maximum (polynomial) non trivial. Nous évoquons ensuite les limites de l' approche GT telle que représentée par le CKPP.

La section 2.1 définit le problème standard du routage (GT) sous la forme d'un problème d'optimisation combinatoire qui sera mis en relation avec les problèmes de

plus courts chemins de la littérature. Dans la section 2.2, nous présentons sa formulation sous la forme d'un programme linéaire en nombres entiers. Dans la section 2.3, nous étudions la complexité du problème en relation avec les problèmes de la littérature pour mettre en évidence les difficultés qui sont inhérentes à sa résolution. Dans la section 2.4, nous étudions la tâche de spécification de flots de communication valides, pour l'application et comme donnée d'entrée du CKPP, qui se ramène à la résolution d'un problème de flot maximum. Dans la section 2.5, nous explorons les limites de l'approche et proposons d'y introduire de la reconfiguration dynamique. Nous terminerons par la conclusion.

2.1 Définition du problème standard

2.1.1 Contexte et notations

Pour mieux cerner le problème de routage à garantie de trafic (GT), dont nous comptons donner une formulation sous la forme d'un problème d'optimisation combinatoire, il convient de préciser le contexte du problème et de donner quelques définitions utiles pour la suite.

Pour mémoire, un NoC est constitué d'un ensemble interconnecté de composants appelés IP. Ces composants IP peuvent être des processeurs, des blocs DSP, ou des blocs de mémoire dans une seule puce. Ils sont des émetteurs et/ou récepteurs de messages. Ces composants sont reliés par des routeurs selon une topologie de réseau donnée. Pour rendre les choses concrètes, nous pouvons examiner l'exemple du NoC spécifié à la figure 2.1. Le NoC est composé de 9 routeurs, notés $R0$ à $R8$, interconnectés suivant la topologie des liens sur la figure. Aux différents routeurs, sont connectés 7 modules IP correspondant à des unités de calcul ou de stockage. Ces composants sont notés $IP_1, \ldots, IP_4$ et $M_1, \ldots, M_3$ sur la figure. La nature de ces unités n'a pas ici d'importance si ce n'est le fait de savoir qu'elles sont émettrices et/ou réceptrices de messages, elles sont appelées également terminaux. Ici, $K = 28$ messages sont spécifiés et émis avec un temps de cycle $T = 8$ (voir plus loin). Ces unités communiquent avec le réseau de routeurs via des interfaces standardisées appelées « interface réseau » (NI). Un IP est exclusivement connecté à un seul routeur via une interface réseau (NI). Ce sont les NI qui sont chargées de préparer le formatage des messages et paquets avant injection dans le réseau. Aussi, par la suite nous utilisons parfois le terme de NI de manière interchangeable pour parler des composants terminaux IP. Plus formellement, un NoC peut être modélisé par un graphe orienté $G = (V, A)$, où l'ensemble V des sommets représente les routeurs et les IP, et l'ensemble des arcs $A \subseteq V \times V$ représente les liaisons de transmission orientées entre les sommets. Ici, nous considérons un routage *wormhole* et un chemin comme une séquence de sommets noté $v_0, v_1, \ldots, v_n$, avec $v_i \in V$. La définition et la notation ainsi données du chemin seront utilisées tout au long de ce manuscrit. Ainsi, chaque message μ est une séquence de q paquets contiguës, avec $q > 0$, qui sont transmis le long des arcs du réseau. Le paquet d'entête contient la

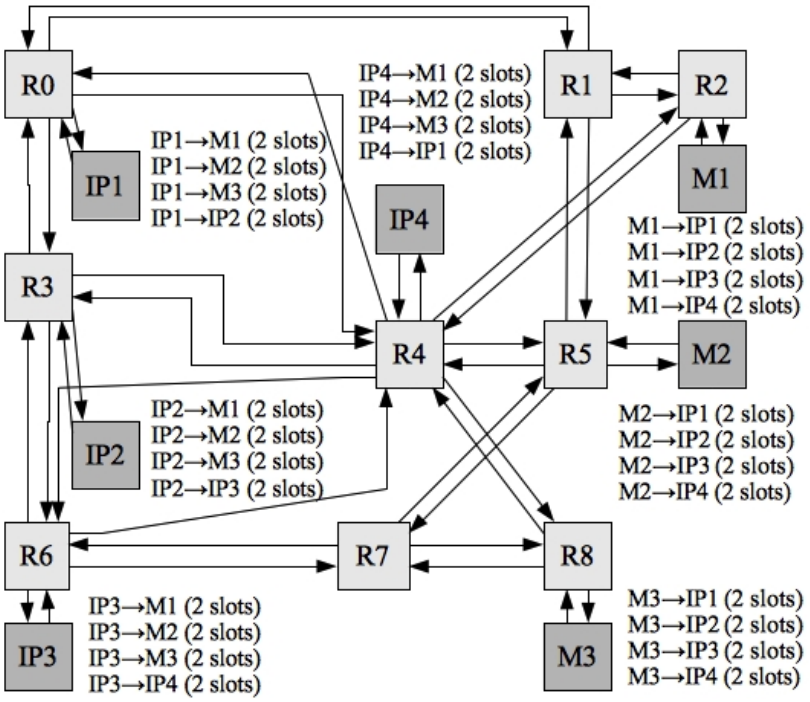


Figure 2.1: Jeux de test N2, avec $T = 8, R = 9, NI = 7, K = 28$.

spécification du chemin origine/destination, tandis que les autres paquets contiennent les données du message. Un routeur n'a pas de capacité de mémorisation de paquet, il retransmet les paquets tels que spécifiés par le paquet d'entête. Le NoC est synchrone, il est cadencé par une horloge commune partagée par tous ses composants. Chaque arc a une capacité d'un paquet par unité de temps ou (*time-slot*). Les paquets sont transmis d'une manière contiguë. Par conséquent, si une occurrence physique d'un message émis à l'instant $t(\mu)$ suit le chemin $v_0, v_1, \ldots, v_n$, avec $v_i \in V$, ses paquets franchiront l'arc (v_i, v_{i+1}) aux *time-slots* consécutifs $t(\mu)+i+k, k = 0, \ldots, q-1$. Nous disons qu'un arc est « occupé » par un paquet à un *time-slot* donné lorsqu'il est franchi à ce *time-slot*, il est dit « libre » sinon à ce même instant. Nous considérons que les chemins peuvent contenir des circuit lorsque cela est nécessaire et ne nous restreignons nullement à des chemins élémentaires. Pour permettre le partage des liens, nous adoptons le mécanisme d'accès multiple à répartition dans le temps par l'utilisation de tables TDMA. Pour chaque nœud émetteur, une table TDMA spécifie l'instant de départ de chaque message à l'intérieur de l'intervalle $[0, \ldots, T-1]$ en tenant compte de la taille de chaque message. Un slot de temps est alloué à chaque paquet émis. Les messages sont émis périodiquement avec une période de longueur T time-slots. Ainsi, le nombre de paquets par message définit une bande passante entre le nœud source et le nœud destination. Par conséquent, un message peut être vu comme la classe de ses occurrences physiques à chaque cycle.

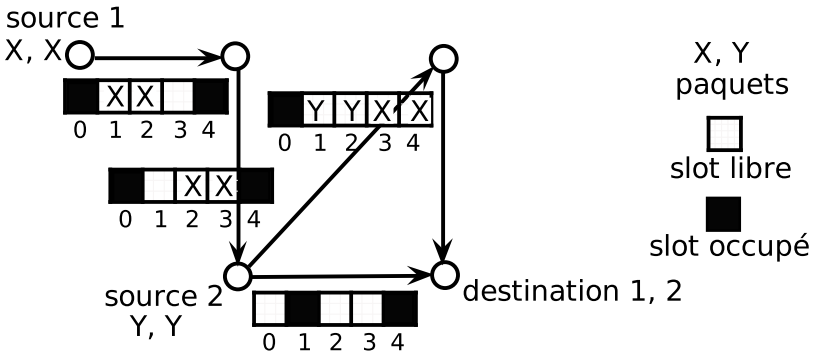


Figure 2.2: Transfert de deux messages dans un NoC occupé.

Cela permet de considérer des classes de *time-slots* spécifiant l'occupation d'un arc aux différents instants t modulo T. Si un arc est occupé au *time-slot* t, il est considéré occupé également aux *time-slots* $t + kT$, pour tout $k \geq 0$. Nous disons qu'un arc est occupé à $t \bmod T$ pour exprimer son occupation récurrente par un paquet. Ainsi la plupart du temps, nous parlons de paquet, ou de message, pour nous référer à la classe qu'il représente, et d'occurrence physique pour nous référer à un paquet particulier à un instant particulier.

Lorsqu'ils sont transmis dans le réseau, deux paquets réels sont dits sans conflits s'ils ne se rencontrent jamais, c'est-à-dire s'ils n'empruntent jamais le même arc au même *slot* de temps. Par extension, des chemins ou des messages sont dits « sans conflit », ou « sans contention », lorsque leurs paquets n'entrent jamais en conflit deux à deux, qu'ils aient été émis à un même cycle ou à des cycles différents. Puisque les chemins peuvent se partager les ressources communes de communication du réseau, et pour assurer un trafic garanti (GT), les chemins et les dates d'émission doivent être spécifiés de telle sorte que deux paquets ne se rencontrent jamais. La figure 2.2 illustre deux chemins sans conflits dans un NoC déjà occupé, avec un cycle d'émission de longueur 5.

Cette technique d'accès multiple à répartition dans le temps a été déjà présentée et utilisée dans [47, 46, 18, 27, 44, 63]. Ces auteurs ont proposés des méthodes heuristiques pour résoudre le problème. Néanmoins, il est difficile de trouver une formulation parfaitement standardisée du problème car les données du problème sont souvent implicites dans les descriptions des architectures de NoC fournies par les auteurs. Notamment, l'absence de jeux de tests communs à ces approches rend difficile leur évaluation comparative. Nous tentons ici de donner une version bien formalisée du problème et par la suite de proposer un cadre d'évaluation et des jeux de tests entièrement reproductibles. Cette formalisation a aussi pour but de faciliter l'étude de la complexité du problème en relation avec les autres problèmes classiques de routage.

2.1.2 Définition

A partir des notations de la section précédente, nous pouvons maintenant énoncer le problème d'optimisation combinatoire modélisant le routage (GT) et que nous appelons « problème cyclique des K-plus-courts chemins sans conflits » ou *Cyclic K-conflict-free shortest Paths Problem* (CKPP).

- *Problème cyclique des K-plus-courts chemins sans conflits (CKPP)*
  Une instance du problème consiste en un graphe orienté $G = (V, A)$, un cycle d'émission de longueur T, et un ensemble de K messages
  $M = \{\mu_i = (s_i, t_i, q_i) : s_i \in V, t_i \in V, q_i > 0, i = 1, \ldots, K\}$, où (s_i, t_i) est une paire origine/destination, et q_i le nombre de paquets du message. Le but est de trouver les instants de départ des messages $t(\mu_i) \in [0, T-1]$, $\mu_i \in M$, à chaque nœud source et les chemins origine/destination non conflictuels dans le graphe de longueur totale minimale.

Il convient de noter que l'objectif du problème est la minimisation de la longueur totale des chemins, tandis que les contraintes résident dans l'occupation temporelle des arcs tels que les chemins soient sans conflits.

2.1.3 Version min-sum et min-max

Le CKPP que nous avons défini et formulé suivant l'approche standard du routage (GT) possède par défaut pour fonction objectif la minimisation de la longueur totale des chemins. Cet objectif permet de minimiser la longueur totale des liens physiques parcourus par des paquets et donc de minimiser la déperdition d'énergie totale induite par le cheminement dans le réseau et le franchissement des routeurs. Nous présentons cette version standard comme la version *min-sum* du problème. Une deuxième version du problème que nous considérons, consiste à prendre pour fonction objectif la minimisation de la longueur du chemin le plus long. Cette version, que nous appelons version *min-max*, prend ainsi en compte la minimisation de la latence maximum d'une communication et permet d'obtenir indirectement un codage du chemin plus réduit. Nous étudierons dans les expérimentations la prise en compte de chacun de ces deux objectifs possibles pour le problème. Par défaut nous parlons de la version *min- sum* du problème.

2.2 Modèle linéaire en nombre entiers

Nous proposons ici un modèle linéaire en nombre entiers ILP ou *integer linear programing model* pour le CKPP [10]. C'est un modèle orienté paquets. Cela signifie que le routage porte sur les paquets. Un message, correspondant à une communication, est divisé en paquets. Le nombre de paquets de chaque message définit sa taille. Nous donnons ci-après les variables, contraintes et objectif du problème.

2.2.1 Variables

Le NoC est composé de routeurs et de paquets qui le traversent en empruntant les arcs. Plus précisément, les données et variables sont les suivantes :

- Il y a n routeurs désignés par i ou j. L'ensemble des routeurs est $R = \{0, \ldots, n-1\}$;
- Il y a m arcs orientés dénommés (i, j). L'ensemble des arcs est désigné par A;
- Il y a K paquets désignés par k avec $k \in \{1, \ldots, K\}$;
- Chaque paquet a un routeur origine $o_k \in R$ et un routeur destination d_k avec $o_k \neq d_k$;
- Le temps est discrétisé, $t \in \{0, \ldots, T-1\}$ est un slot de temps, et T est la taille de la TDMA;
- Le système est pipeliné, les slots de temps t et $t+T$ sont identiques pour tout t;
- La variable $x_{i,j,k,t}$ est une variable booléenne qui est égale à 1 si le paquet k utilise l'arc (i, j) durant le slot de temps t, et zéro sinon.

2.2.2 Contraintes

Les contraintes sont les suivantes :

1. **Capacités des arcs.** Chaque arc ne peut acheminer plus d'un paquet par unité de temps.

$$\sum_{k \in \{1,\ldots,K\}} x_{i,j,k,t} \leq 1 \quad \forall (i,j) \in A, \forall t \in \{0, \ldots, T-1\} \tag{2.1}$$

2. **Origine des paquets.** Pour tout k dans $\{1, \ldots, K\}$, le paquet k a pour origine le routeur o_k.

$$\sum_{\substack{t \in \{0,\ldots,T-1\} \\ i \in R | (o_k, j) \in A}} x_{o_k,j,k,t} = 1 \quad \forall k \in \{1, \ldots, K\} \tag{2.2}$$

3. **Destination des paquets.** Pour tout k dans $\{1, \ldots, K\}$, le paquet k a pour destination le routeur d_k.

$$\sum_{\substack{t\in\{0,\ldots,T-1\}\\ i\in R|(i,d_k)\in A}} x_{i,d_k,k,t} = 1 \quad \forall k \in \{1,\ldots,K\} \tag{2.3}$$

4. **Conservation des paquets.** Pour tout instant t dans $\{0,\ldots,T-1\}$, pour tout k dans $\{1,\ldots,K\}$ et pour tout i dans $R\backslash\{o_k,d_k\}$, si le paquet k atteint le routeur i à l'instant t, alors il devrait avoir quitté ce routeur à l'instant $t+1$.

$$\sum_{j\in R|(j,i)\in A} x_{j,i,k,t} = \sum_{j\in R|(i,j)\in A} x_{i,j,k,(t+1) \bmod T}$$
$$\forall t \in \{0,\ldots,T-1\}, \forall k \in \{1,\ldots,K\}, \forall i \in R\backslash\{o_k,d_k\} \tag{2.4}$$

5. **Pas de retour au routeur d'origine.** Pour tout instant t dans $\{0,\ldots,T-1\}$, pour tout k dans $\{1,\ldots,K\}$, le paquet k ne peut retourner au routeur o_k après l'avoir quitté.

$$\sum_{i\in R|(i,o_k)\in A} x_{i,o_k,k,t} = 0 \quad \forall t \in \{0,\ldots,T-1\}, \forall k \in \{1,\ldots,K\} \tag{2.5}$$

6. **Pas de sortie du routeur destination.** Pour tout instant t dans $\{0,\ldots,T-1\}$, pour tout k dans $\{1,\ldots,K\}$, le paquet k ne peur ressortir du routeur d_k après l'avoir atteint.

$$\sum_{j\in R|(d_k,j)\in A} x_{d_k,j,k,t} = 0 \quad \forall t \in \{0,\ldots,T-1\}, \forall k \in \{1,\ldots,K\} \tag{2.6}$$

7. **Les routeurs d'origine génèrent au plus un paquet par unité de temps.** Pour tout slot t dans $\{0,\ldots,T-1\}$, pour tout routeur $i \in R$, le nombre total de paquets émis par i tel que $o_k = i$ doit être au plus un.

$$\sum_{\substack{k\in\{1,\ldots,K\}|o_k=i\\ j\in R|(i,j)\in A}} x_{i,j,k,t} \leq 1 \quad \forall t \in \{0,\ldots,T-1\}, \forall i \in R \tag{2.7}$$

8. **Les routeurs destination consomment au plus un paquet par unité de temps.** Pour tout slot t dans $\{0,\ldots,T-1\}$, pour tout routeur $j \in R$, la somme des paquets k envoyée au routeur j et tels que $d_k = j$ doit être au plus un.

$$\sum_{\substack{k\in\{1,\dots,K\}|d_k=j \\ i\in R|(i,j)\in A}} x_{i,j,k,t} \leq 1 \quad \forall t \in \{0,\dots,T-1\}, \forall i \in R \tag{2.8}$$

9. **Contraintes de Message.** Un message est modélisé comme une séquence de paquets qui doivent avoir la même route dans le NoC. Plus formellement, le message $\mu \in M$ est un ensemble de paquets ordonnés $\{\omega_{\mu,1},\dots,\omega_{\mu,q}\}$ où q est la longueur du message μ (c'est-à-dire le nombre de paquet du message), $\omega_{\mu,1} \in \{1,\dots,K\}$ est le premier paquet et $\omega_{\mu,l(\mu)}$ est le dernier paquet du message μ. Tous les paquets d'un message doivent avoir le même routeur origine et le même routeur destination, c'est-à-dire ils doivent satisfaire: $o_k = o_{k'}$ et $d_k = d_{k'} \quad \forall \mu \in M, \forall k,k' \in \{\omega_{\mu,1},\dots,\omega_{\mu,q}\}$. Les contraintes de message sont représentées comme suit :

$$x_{i,j,\omega_{\mu,p},t} = x_{i,j,\omega_{\mu,p+1},(t+1) \bmod T} \quad \forall \mu \in M, \forall p \in \{1,\dots,q-1\}, \forall t \in \{0,\dots,T-1\}, \forall (i,j) \in A \tag{2.9}$$

Ces contraintes indiquent que si un paquet $\omega(\mu,p)$ occupe l'arc (i,j) au temps t, alors le paquet $\omega(\mu,p+1)$ doit occuper cet arc au temps $(t+1) \bmod T$. L'égalité implique aussi que si le paquet $\omega_{\mu,p}$ n'a pas occupé l'arc (i,j) au temps t, alors le paquet $\omega_{\mu,p+1}$ ne peut occuper cet arc au temps $(t+1) \bmod T$. Ainsi, tous les paquets d'un message se déplacent le long de la même route au cours du temps.

2.2.3 Objectif

La fonction objectif consiste ici à minimiser la somme des longueurs des chemins, elle est donnée par :

10. **Minimisation de la somme des longueurs**

$$Minimiser \sum_{(i,j)\in A} \sum_{k\in K} \sum_{t\in T} x_{i,j,k,t} \tag{2.10}$$

La longueur d'un chemin s'exprime en nombre de slots de temps. La longueur totale des chemins correspond donc au nombre de slots de temps nécessaires pour convoyer tous les messages à leur destination. La fonction consiste donc à minimiser ce nombre.

2.3 Complexité

Dans cette section, nous relions le CKPP à d'autres problèmes classiques de la littérature afin d'évaluer sa complexité. Nous montrons que le problème est NP-difficile au sens

fort parce qu'étant une combinaison et une extension d'un problème de *Bin Packing* et d'un problème de chemins disjoints. Également, nous relions le problème à des problèmes de flot. Certains de ses sous-problèmes, où l'on cherche un unique chemin origine-destination, ont des connexions avec des problèmes de plus court chemin avec fenêtre de temps qui peuvent être résolus avec un algorithme pseudo-polynomial. Pour commencer, nous énonçons d'abord le théorème suivant:

Théorème 1. *Le CKPP est NP-difficile au sens fort.*

Preuve La preuve est faite par réduction d'un problème de *Bin Packing* au CKPP. Puisque le *Bin Packing* est NP-complet au sens fort, nous montrons qu'il existe une transformation polynomiale vers le CKPP [21]. Dans le *Bin Packing*, on donne un ensemble $\{x_1, x_2, \ldots, x_n\}$ d'objets, chaque objet x_i ayant une taille q_i, et k *bins* de capacité B. La question est de savoir si l'on peut attribuer tous les objets aux *bins* de telle manière qu'aucun *bin* ne reçoive des objets dont la taille totale excède B. La construction d'une instance du CKPP à partir d'une instance d'un *Bin Packing* consiste à construire un graphe pour envoyer n messages des nœuds origines o_i distincts vers des nœuds destination d_i distincts en traversant une coupe minimale du graphe avec k arcs, avec un cycle d'émission de longueur B. Chaque message est composé de q_i paquets. Cette transformation est illustrée dans la figure **??** avec différentes topologies possibles de NoC. La topologie peut être générale comme dans (a) ou planaire comme dans (b) et (c). La figure montre une transformation d'un *Bin Packing* avec 2 *bins* (taille de la coupe) et 3 objets (nœuds terminaux). Un point important à souligner est que tout message peut être acheminé à partir de son nœud origine à son nœud destination par n'importe quel arc de la coupe, suivant un instant de départ pris dans $[0, B-1]$. Nous pouvons noter que de tels chemins ont une longueur totale minimale. Par conséquent, il existe n chemins sans conflits (de longueur donnée), avec des dates d'émission des messages associés dans $[0, B-1]$, si et seulement si l'instance du *Bin Packing* a une solution. Étant donné que $q_i \leq B$ et $k \leq n$, il s'ensuit que la transformation peut être effectuée en un temps polynomial avec la taille du problème d'entrée. Il en découle que le CKPP est NP-difficile au sens fort. smartqed

La réduction ici du *Bin Packing* de la figure 2.3 en un CKPP (2 *bins*, 3 objets), consiste à acheminer les messages d'origine o_1, o_2 et o_3 respectivement aux destinations d_1 de taille q_1 , d_2 de taille q_2 et d_3 de taille q_3. Tous les messages devant obligatoirement passer par les deux arcs de la coupe minimale à des instants compris dans $[0, B-1]$.

Il convient de noter que le CKPP est NP-difficile au sens fort aussi bien pour des graphes généraux, des graphes planaires ou des réseaux en grille. Le qualificatif « fort » indique que le problème demeure NP-difficile même pour une « petite » taille du cycle d'émission T. Nous entendons par « petite » , « polynomialement borné » par rapport à la taille de la donnée d'entrée. Nous ne pouvons pas établir que le problème est NP complet, car il n'est pas clair si le CKPP est dans NP. La difficulté est que la longueur d'un chemin peut être aussi grande que $N \times T$, pour un graphe de taille N avec un cycle d'émission de longueur T. La taille de la donnée d'entrée

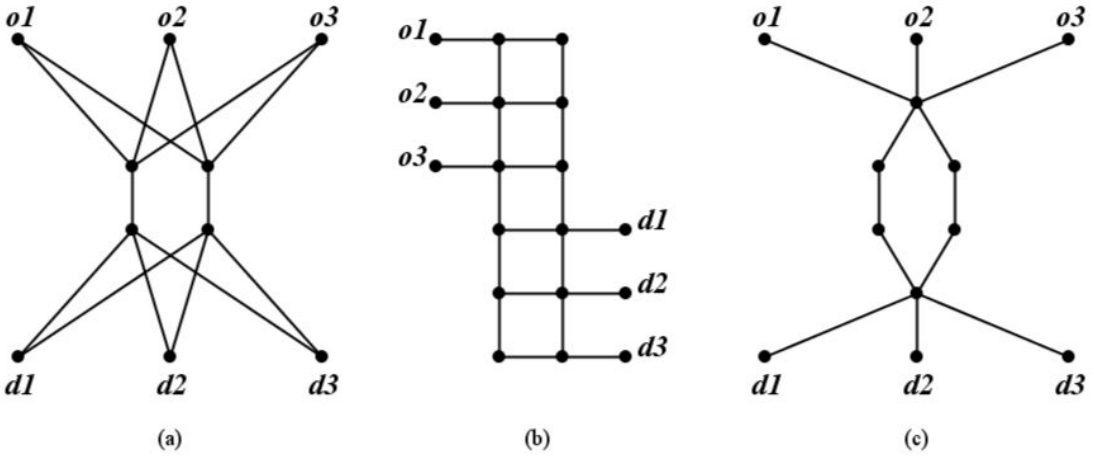


Figure 2.3: Réduction d'un *Bin Packing* en un CKPP (2 *bins*, 3 objets).

est de $\mathcal{O}(k \times \log(T) + N + \log(T))$ où k est le nombre de paires origine/destination. Par conséquent, la longueur de la chaîne codée nécessaire pour représenter le chemin pourrait ne pas être polynomiale avec la taille de la donnée d'entrée.

Une autre façon de voir que le CKPP est NP-difficile au sens fort, consiste à le rapprocher du problème de plus courts chemins disjoints (EDP*) ou de chemins disjoints (*K-Disjoint Paths problem*) [35], si l'on ne tient pas compte de la minimisation de la longueur des chemins. Ce problème peut se décliner en différentes versions selon que nous voulons des chemins de sommets ou d'arcs disjoints reliant un ensemble de K paires source-destination. C'est l'un des premiers problèmes NP-difficiles rapportés par Karp [33]. En ramenant le CKPP aux seules instances pour lesquelles $T = 1$, sa restriction devient un problème de plus courts chemins disjoints de type EDP. Ce problème est connu pour être NP-difficile également dans le cas de graphes planaires [45], ou une topologie en grille [22]. Puisque, le CKPP demeure NP-difficile même lorsque le nombre T est borné par une constante, il s'ensuit qu'il est NP-difficile au sens fort.

Un problème similaire est le problème de flot insécable UFP. C'est une généralisation de l'EDP où chaque arête e a une capacité positive c_e; et chaque paire a une demande $d_i > 0$. La demande de s_i à t_i doit être acheminée de manière insécable, c'est-à-dire le long d'un chemin unique de s_i à t_i. Pour chaque arête e la demande totale acheminée par ce lien doit être au plus égale à c_e. Ce problème ajoute une contrainte de capacité à l'EDP. Il est différent du CKPP dans lequel la capacité des arcs est définie par une occupation temporelle et cyclique des arcs sans conflit avec des instants spécifiques de passage.

Généralement, les modèles classiques de problème de flot portent sur une situation statique. Ils ne sont souvent pas appropriés pour prendre en compte toutes les

variétés d'applications dans lesquelles les valeurs des flots sur les arcs varient avec le temps ou nécessitent une certaine durée pour l'acheminement via chaque arc. Un tel problème où sont pris en compte des temps de transit des flots est représenté par le problème de plus court chemin avec fenêtres de temps ou *one-to-one shortest path problem with time windows* (SPPTW) [13, 48]. Étant donné un graphe G, un nœud source s, un nœud de destination t, un instant de départ θ, un temps de transit τ_a, un coût c_a et un ensemble de fenêtres de temps $\mathcal{F}_a$ sur chaque arc a. L'objectif est de calculer le plus court chemin (concernant les coûts c_a) respectant les fenêtres de temps données. Ce genre de problème se rencontre souvent dans les transports terrestres, le contrôle du trafic routier, et dans des applications de routage de véhicules. Le problème est NP-difficile, mais il est résolu de façon exacte par plusieurs algorithmes pseudo-polynomiaux [13, 29, 56].

Considérant le CKPP, un de ses sous problèmes où l'on cherche un chemin origine/destination unique dans un NoC déjà occupé est semblable à un problème de routage origine/destination avec fenêtres de temps. Dans ce cas, une approche de résolution consiste à utiliser un graphe spatio-temporel étendu (TEG) [34]. Le TEG contient une copie de l'ensemble des nœuds et des arcs du graphe initial pour chaque slot de temps considéré. Un plus court chemin sans conflit dans le NoC peut être considéré comme un plus court chemin statique dans le graphe spatio-temporel étendu. Ce sous problème diffère du SPPTW par son caractère cyclique avec des fenêtres de temps, capacités et durées de transfert égaux à l'unité. Nous présentons dans ce document un algorithme Dijkstra modifié permettant de calculer un chemin origine-destination dans le TEG. Cet algorithme est pseudo polynomial avec la taille du CKPP. Il sera utilisé comme opérateur dans les heuristiques et métaheuristiques présentées dans ce travail pour traiter l'ensemble du CKPP.

2.4 Spécification des flots de communication

Le CKPP formalise un problème de routage dans un NoC représenté par un graphe orienté $G = (V, A)$, un cycle d'émission de longueur T, et un ensemble de K messages $M = \{\mu_i = (s_i, t_i, q_i) : s_i \in V, t_i \in V, q_i > 0, i = 1, \ldots, K\}$, où (s_i, t_i) est une paire origine/destination, et q_i le nombre de paquets du message. Il porte donc sur des flots de communication définis par les messages dont les bandes passantes sont données par les quantités de paquets. Ces flots insécables se partagent une bande passante de taille maximum T qui définit la capacité des liens du réseau. Chaque IP émetteur ou récepteur étant relié à son routeur via un lien unique, il convient de s'assurer du respect de la capacité de ce lien en entrée et en sortie du réseau et cela indépendamment de la topologie du réseau. Dans [9], la spécification des flots de communication est basée sur l'analyse des graphes de dépendance de communication de l'application et doit aboutir à une répartition équitable des bandes passantes conforme aux besoins de l'application. Ici, nous formalisons le problème de la construction de ces flots de données valides.

Nous la représentons comme la résolution d'un problème de flot maximum standard ce qui permet d'en évaluer la difficulté. Également, nous en déduirons un algorithme de génération de jeux de données aléatoires présenté plus loin dans la thèse. Ces jeux de données seront utilisés pour les évaluations des algorithmes de résolution du CKPP.

Soit $\Pi = \{1, 2, \dots P\}$ l'ensemble des IP, pouvant être des nœuds émetteurs ou des nœuds récepteurs. Nous supposons donné le graphe de communication de l'application (ACG*) qui précise les flots de communication ouverts entre IP. Il est représenté par une matrice $C = (c_{ij})$, $1 \leq i, j \leq P$, telle que $c_{ij} \in \{0, 1\}$. Une entrée c_{ij} est telle que $c_{ij} = 1$ si une communication est ouverte i à j, $c_{ij} = 0$ sinon. Notamment nous avons $c_{ii} = 0$ car une IP ne communique pas avec elle-même. Nous notons x_{ij} la variable indiquant le nombre de paquets associés à chaque liaison origine/destination (i, j). Nous spécifions par s le nombre minimum de paquets émis par une source donnée. La problématique de la génération de flots de communication valides, que nous intitulons problème d'injection de trafic (TIP*), consiste à générer rapidement des solutions optimales ou quasi-optimales du problème de flot maximum suivant:

$$maximiser \sum_{i,j \in \{1,\dots,P\}} c_{ij} x_{ij} \tag{2.11}$$

sous les contraintes

$$\sum_{i \in \{1,\dots,P\}} c_{ij} x_{ij} \leq T, \quad \forall j \in \{1, \dots, P\} \tag{2.12}$$

$$\sum_{j \in \{1,\dots,P\}} c_{ij} x_{ij} \leq T, \quad \forall i \in \{1, \dots, P\} \tag{2.13}$$

$$c_{ij} x_{ij} \geq c_{ij} s, \quad \forall i, j \in \{1, \dots, P\} \tag{2.14}$$

$$x_{ij} \in \{0, \dots, T\}. \tag{2.15}$$

La quantité $\sum_{i,j} c_{ij} x_{ij}$ de la relation (2.11) est la fonction objectif qui correspond à la maximisation du flot total qui est injecté dans le réseau durant la période T. La relation donnée par l'inéquation (2.12) exprime le fait que le nombre de paquets émis par une source vers toutes ses destinations possibles, pendant un cycle de longueur T, ne peut excéder T. L'inéquation (2.13) exprime que le nombre de paquets en provenance des nœuds sources et arrivant à un nœud destination, durant un cycle T, ne peut être plus grand que T. La relation (2.14) indique que le nombre de paquets émis est supérieur au seuil s. Par la suite, nous évaluons le niveau de saturation des émetteurs, appelé (TSL*), comme la quantité

$$TSL = \left(\sum_{i,j} c_{ij} x_{ij} / P_E T \right) \times 100, \tag{2.16}$$

où P_E est le nombre d'IP émetteurs. Le TSL définit le pourcentage de paquets émis

relatif aux slots de temps disponibles par rapport à chaque nœud émetteur. Un TSL de 100% indique que chaque émetteur produit exactement un paquet à chaque slot de temps. Cette quantité est souvent dénommée dans la littérature des SoC par l'expression *message throughput* [54]. La figure 2.4 illustre le problème de la génération de trafic dans un cas simple. Un ACG est donné en (a), tandis que (b) et (c) présentent deux configurations optimales possibles de flots de trafic. Il convient de noter que, avec un seuil de $s = 0$, des solutions optimales peuvent être peu satisfaisantes, car certains IP n'ont pas de paquets à émettre, comme en (b). Il peut être souhaitable de mieux partager et décomposer le flot entre les émetteurs/récepteurs de manière plus équitable, comme en (c). Il est important de noter que la valeur du flot maximum dépend d'abord de la structure du ACG, indépendamment de la topologie du NoC. A titre d'exemple, le TSL ne peut pas dépasser 66,6% avec l'ACG de la figure 2.4. Cet exemple illustre comment les contraintes de bandes passantes de l'approche GT doivent être soigneusement ajustées.

	IP0	IP1	IP2
IP0	0	1	0
IP1	1	0	1
IP2	0	1	0

(a)

	IP0	IP1	IP2	Total
IP0	0	T	0	T
IP1	0	0	T	T
IP2	0	0	0	0
Total	0	T	T	2T

(b)

	IP0	IP1	IP2	Total
IP0	0	T/2	0	T/2
IP1	T/2	0	T/2	T
IP2	0	T/2	0	T/2
Total	T/2	T	T/2	2T

(c)

Figure 2.4: Un ACG (a), et deux configurations de flot maximum (b-c).

2.5 Limites de l'approche avec garantie de trafic

La technique d'accès multiples à répartition dans le temps permet de garantir un débit en trafic donné. Résoudre le CKPP doit permettre de garantir une latence minimum des communications et un routage exempt de conflit et donc d'interblocage. Le routage à la source permet l'absence de conflit et autorise également l'emploi de routeurs très simples et sans buffer de mémoire. Néanmoins, dans le routage (GT), l'allocation des ressources de communication est effectuée pour des cas extrêmes. Le dimensionnement du réseau peut être fonction de communications très exigeantes en bande passante alors que cette exigence est parfois ponctuelle. Dans la figure 2.5, la bande passante du nœud source IP i est repartie de façon définitive entre trois nœuds destination. Le nœud destination IP j reçoit une partie Xij de cette bande passante. Il ne tient pas compte de la fluctuation à la baisse des besoins en bande passante qui peut apparaître

lors du fonctionnement. Il s'ensuit un sur- dimensionnement possible du réseau et une utilisation sous-optimale de ses ressources physiques. Comme illustré par la figure 2.5, un désavantage majeur du routage (GT) tel que modélisé par le CKPP est qu'il fige de façon définitive la bande passante entre la source et la destination d'une communication. Nous proposons d'atténuer ce désavantage par la possibilité d'une modification dynamique des flots de données. Nous présentons cette problématique dans le chapitre suivant et en déduisons une version remaniée du problème de routage.

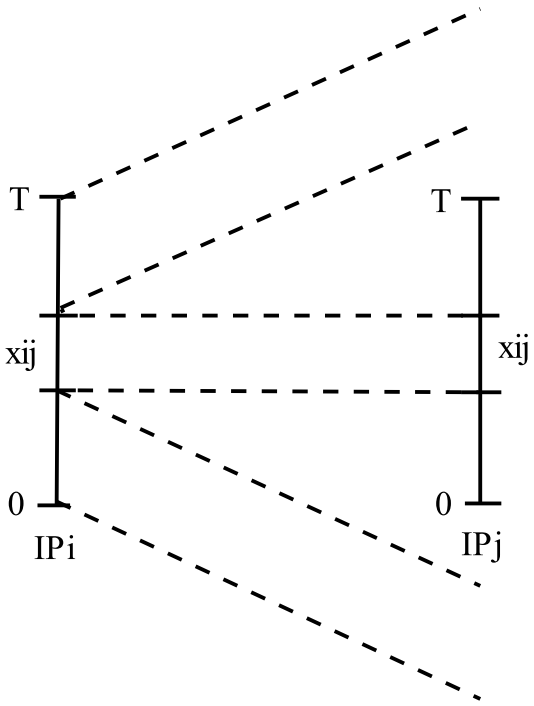


Figure 2.5: Bande passante figée entre la source i et la destination j.

2.6 Conclusion

Nous avons présenté le problème de routage à garantie de trafic (GT) selon le mécanisme d'accès multiple à répartition dans le temps par l'utilisation de tables TDMA. Cela nous a permis de mettre en évidence un certain nombre de ses caractéristiques et les difficultés qui lui sont propres. Nous avons formalisé la problématique par le biais d'un problème d'optimisation combinatoire (NP-difficile au sens fort) de calcul de plus courts chemins avec contraintes, nommé problème cyclique des K-plus-courts chemins sans conflits (CKPP). Il peut être considéré sous sa version *min-sum* ou *min-max*. Sa formulation sous la forme d'un programme linéaire en nombres entiers a été donnée afin d'aborder sa résolution de manière exacte par des solveurs standards. Nous avons

présenté la spécification des flots de communication donnés en entrée du CKPP comme la résolution d'un problème de flot maximum entre IP émetteurs et récepteurs. Suivant les contraintes de l'application, le concepteur est ainsi amené à proposer des solutions sous-optimales de ce problème particulier répondant aux besoins en bande passante. La limite principale de cette modélisation du trafic GT réside dans le manque de flexibilité des bandes passantes qui sont figées une fois pour toute, entraînant un sur-dimensionnement du réseau pour satisfaire des conditions au pire cas. Nous proposons maintenant d'aborder le problème de la reconfiguration dynamique des bandes passantes des communications.

3

Problème de routage reconfigurable avec garantie de trafic

Le problème d'optimisation combinatoire formulé dans le chapitre 2, est un problème de routage avec contraintes dans lequel des bandes passantes sont spécifiées une fois pour toute via les définitions des tables TDMA. L'une des faiblesses de cette approche est le comportement statique qu'elle confère au NoC, le rendant dépendant d'un débit fixé à l'avance et surdimensionné pour traiter les pires cas de trafic. Pour atténuer cette contrainte nous proposons ici une extension du premier problème. Dans cette extension, les bandes passantes peuvent être modifiées dynamiquement. Et cela, par le changement en cours de fonctionnement d'une ou de plusieurs TDMA utilisées par les émetteurs. L'émetteur possède des TDMA dont le nombre est fixé à l'avance. Mais, leurs utilisations peuvent être interverties à tout moment à chaque début de cycle d'émission, selon le choix propre à un émetteur donné. Il faut alors assurer une compatibilité des TDMA lors de leurs permutations asynchrones et indépendantes pour garantir l'absence de conflit.

Cela nous conduit à présenter une version du problème de routage GT incluant la reconfiguration dynamique des tables TDMA. Nous appelons ce problème « problème cyclique des K-plus-courts chemins reconfigurables sans conflits » (CRKPP*). De même que pour le cas standard, nous analysons les contraintes que doivent respecter les spécifications de flot de communication pour constituer des données d'entrée valides du problème. Cela doit permettre de préciser les limites de l'approche proposée. Nous proposons de distinguer différents types de reconfigurations des communications possibles, suivant la durée d'observation des scénarii de trafic rencontrés. Cela nous conduit à la problématique du couplage entre trafic *Best Effort* (BE) et trafic garanti (GT) dans un même NoC, en l'occurrence le NoC μSpider II.

La section 3.1 présente le principe de la reconfiguration dynamique de manière informelle sur des exemples et rappelle les notations de base, tandis que la section 3.2 définit le problème précisément. La section 3.3 aborde l'étude des flots de communication valides pour le problème. Les sections 3.4 et 3.5 traitent ensuite respectivement des limites et extensions possibles des mécanismes de reconfiguration et d'adaptation au trafic. Nous terminons par une conclusion.

3.1 Principe de la reconfiguration dynamique

Dans sa version standard présentée au chapitre 2, le problème de routage à garantie de trafic (GT) n'admet qu'une seule et unique configuration possible de bande passante entre un émetteur et un récepteur. Nous voulons maintenant assouplir cette contrainte et rendre le NoC plus adaptatif aux variations de trafic. Il convient de lui permettre de redimensionner ses ressources en communication en cours d'exécution et sans perturbation ni arrêt du fonctionnement du NoC. L'exemple de la figure 3.1 illustre l'intérêt d'une telle possibilité de changement dynamique de la configuration d'émission. Ainsi qu'indiqué par les étapes (a), (b), (c), un IP *i* peut augmenter sa bande passante vers un IP *j* tout en cessant d'émettre vers d'autres IP. Le but est de rendre possible une telle modification dynamiquement selon le choix propre à l'émetteur à chaque début de cycle d'émission sans que cela entraîne de conflit entre paquets.

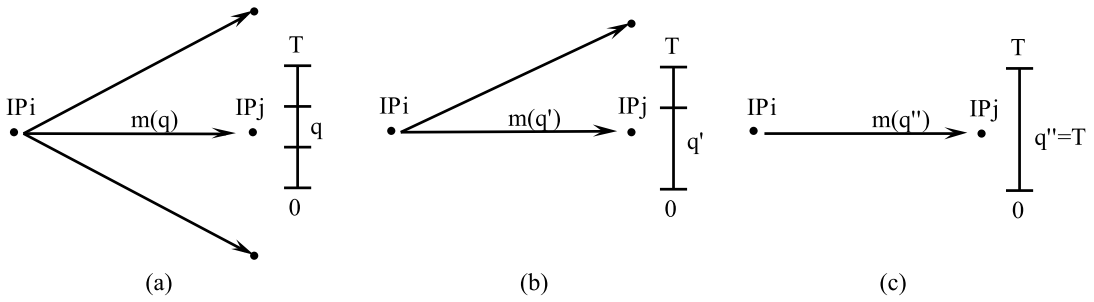


Figure 3.1: Modification du flot d'émission selon le choix de l'émetteur.

En nous basant sur les notations du chapitre précédent, nous considérons maintenant que chaque émetteur possède un ensemble de tables TDMA d'émission toutes basées sur le même cycle de temps T d'émission. Une TDMA δ peut être considérée comme une spécification d'un ensemble de messages avec leurs dates d'émissions dans l'intervalle $[0, T-1]$. L'ensemble des messages est un ensemble de triplets $M_\delta^s = \{\mu_i = (s, t_i, q_i) : t_i \in V, q_i > 0, i = 1, \ldots, k_\delta\}$, où s est le nœud source des k_δ messages ayant pour destinations les nœuds t_i et q_i paquets chacun. Du fait de l'émission cyclique et séquentielle, nous devons avoir $k_\delta \leq \sum q_i \leq T$. De même, étant donné deux messages avec leurs dates d'émissions respectives t_1, $t_2 \in [0, T-1]$ et leur nombre respectif de paquets q_1, q_2, de sorte que $t_1 < t_2$, nous avons nécessairement $t_1 + q_1 \leq t_2$, et $(t_2 + q_2) \bmod T \leq t_1$ lorsque $t_2 + q_2 \geq T$. Nous détaillons dans la section suivante les contraintes précises que doivent respecter les spécifications de messages pour constituer des flots de données valides du problème. Ainsi qu'illustré par la figure 3.2, la reconfiguration dynamique consiste en la possibilité offerte à chaque émetteur de changer de table d'émission d'un cycle à l'autre selon son choix et de manière indépendante et asynchrone relativement aux choix des autres émetteurs. En (a), sont présentées les

tables TDMA affectées à des composants IP. En (b), sont illustrés des échanges possibles des tables d'émission TDMA à chaque cycle.

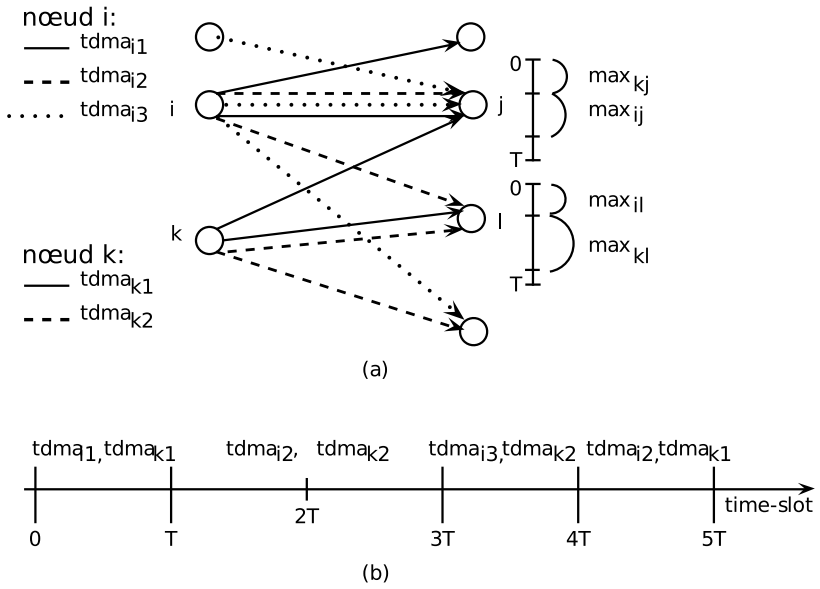


Figure 3.2: Routage reconfigurable. (a) Un émetteur possède des configurations multiples de tables TDMA. (b) Chaque émetteur peut changer de table TDMA d'émission à chaque début de cycle T.

Le but est ici de garantir l'absence de conflit quelle que soit l'application de la table TDMA choisie à chaque cycle d'émission par un émetteur. L'approche de résolution devra tenir compte des différents cas de conflits possibles. Ces conflits peuvent être des conflits inter-messages, c'est-à-dire entre des paquets de messages distincts, ou des conflits intra-messages, c'est-à-dire entre des paquets d'un même message. Par ailleurs, ces conflits peuvent survenir entre messages émis lors d'un même cycle (conflit intra-cycle) ou bien lors de cycles d'émission différents (conflit inter-cycle). On peut noter que les chemins peuvent être de longueur supérieure à T et comporter des boucles. On peut noter également que lors d'un changement de table d'émission au cycle $(k+1)\,T$, tous les paquets liés à la table précédente doivent avoir été émis. En effet, il se peut que des paquets du dernier message émis au cycle précédent kT soient envoyés après le temps $kT + (T-1)$, qui est la date d'envoi la plus tardive pour un paquet d'entête. Il conviendra donc d'assurer l'absence de conflit et permettre une réutilisation maximale des *time-slots* libérés lors d'un changement de table d'émission. Dans la figure 3.3, le message 2 de la tdma 0 de la NI0 est composé de trois paquets et est à destination du routeur R2. Pour y parvenir, il suit le chemin R0, R1, R2. Le message 3 est composé de trois paquets et est à destination du routeur R4. Il suit le chemin R0, R1, R2, R4. Quant à la tdma 1 de la même NI0, est comporte 2 messages. Le premier possède 4

paquets et est à destination du routeur R2 en empruntant le chemin R0, R1, R2. Le deuxième message à destination du routeur R4 est divisé en 5 paquets. Son chemin d'emprunt est le R0, R1, R3, R4. L'exemple de la figure 3.3 illustre des transferts de messages sans conflits dans le NoC avec une NI source possédant deux tables TDMA. Nous pouvons constater comment des chemins origine/destination peuvent réutiliser des slots de temps vacants.

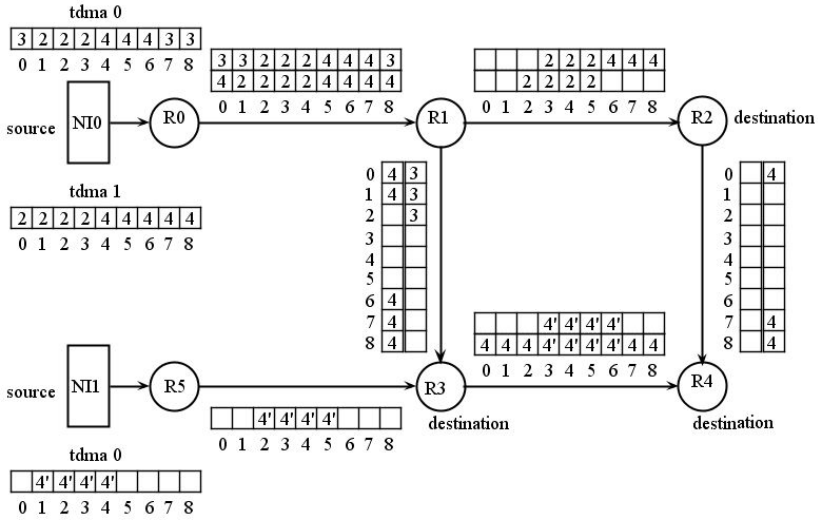


Figure 3.3: Transferts de paquets dans un NoC avec configurations multiples de tables TDMA.

3.2 Définition du problème

Nous pouvons maintenant énoncer l'extension du problème standard au cas reconfigurable. Nous l'appelons « problème cyclique des K-plus-courts chemins reconfigurables sans conflits » CRKPP.

- *Problème cyclique des K-plus-courts chemins reconfigurables sans conflits CRKPP*
  Une instance du problème consiste en un graphe orienté $G = (V, E)$, un cycle d'émission de longueur T, et un ensemble fini de tables TDMA. Un nœud source donné s peut avoir une ou plusieurs tables qui lui sont attachées et qui peuvent être interchangées à chaque cycle d'émission. Une table TDMA δ attachée à un nœud source s est spécifiée par l'ensemble
  $M_\delta^s = \{\mu_i = (s, t_i, q_i) : t_i \in V, q_i > 0, i = 1, \ldots, k_\delta\}$ des messages qui lui sont attachés, où (s, t_i) est une paire origine/destination, et q_i le nombre de paquets. Le but est de spécifier l'instant de départ de chaque message dans l'intervalle $[0, T-1]$ et les chemins origine/destination de longueur totale minimale de routage

des messages, tels que les chemins soient sans conflits quelle que soit la table TDMA qui est émise à chaque cycle.

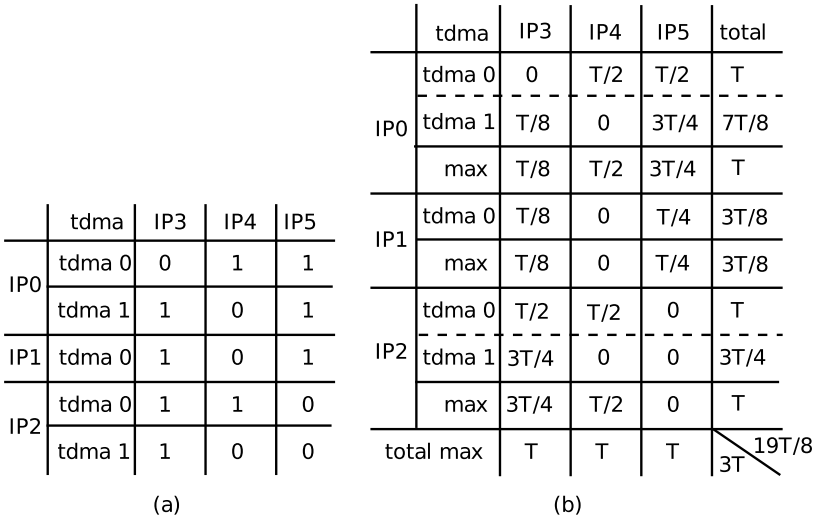
Il convient de noter que l'objectif du problème est soit la minimisation de la somme des longueurs des chemins, soit la minimisation de la longueur du plus long chemin, alors que les contraintes résident dans l'occupation temporelle des arcs telle que les chemins soient sans conflits. Le CRKPP est une extension du CKPP dont on peut définir une version *min-sum*, telle que dans l'énoncé, et une version *min-max* si l'objectif devient la minimisation de la longueur maximum d'un chemin.

3.3 Spécification des flots de communication

Dans le NoC μSpider II [9], la spécification des flots de communication attribués aux différents IP est effectuée à partir de l'analyse de l'application représentée par des graphes de dépendances de communications. Où, chaque graphe représente une tâche. Les échanges entre les tâches représentant les communications. Cette représentation permet de calculer la bande passante réelle, et la taille des données maximale des communications échangées par les composants du NoC. Pour nous, nous partons d'un graphe de communication de l'application qui spécifie les communications souhaitées (ouvertes) entre un ensemble de composants IP, et nous présentons le problème de détermination des valeurs de flot de communication. Il s'agit d'un problème de flot maximum qui étend la version donnée dans le chapitre précédent pour le cas standard. Le but est de donner un éclairage sur la difficulté à élaborer des jeux de données valides pour le problème et illustrer l'intérêt et les limites de l'approche de reconfiguration proposée. Pour nos expérimentations, un de nos objectifs est de générer des jeux de tests aléatoires en grand nombre répondant aux contraintes.

Soit $\Pi = \{1, 2, \dots P\}$, l'ensemble des IP pouvant être soit des nœuds source, soit des nœuds de destination. Pour prendre en compte des configurations d'émission multiples, nous définissons le graphe de communication de l'application (ACG*) comme une matrice multi-lignes ainsi qu'illustré à la figure 3.4-a. Les coefficients de cette matrice $C = (c_{ijk})$, avec $1 \leq i, j \leq P$, $1 \leq k \leq \Theta_i$, et Θ_i le nombre de configurations d'émission (ou tables TDMA) associées au nœud i, sont tels que $c_{ijk} = 1$ si une communication est allouée entre les deux IP i et j, via la table k, $c_{ijk} = 0$ sinon.

En partant de la donnée d'un ACG, il s'agit de maximiser des valeurs de flot x_{ijk} représentant le nombre de paquets associés à chaque liaison source-destination (i, j) et table d'émission k. Sachant que les tables TDMA d'un IP sont interchangeables à chaque cycle d'émission de manière indépendante et asynchrone relativement aux autres émetteurs, il convient de préciser les contraintes qui définissent un flot valide. Afin d'éviter l'introduction de conflits en réception, les différents émetteurs reliés à un récepteur donné (colonne) doivent se partager chacun un montant maximum de trafic. La contrainte sur les émetteurs (lignes) reste inchangée par rapport au cas standard. Nous appelons l'extension du problème de génération de flot standard « problème



	tdma	IP3	IP4	IP5
IP0	tdma 0	0	1	1
	tdma 1	1	0	1
IP1	tdma 0	1	0	1
IP2	tdma 0	1	1	0
	tdma 1	1	0	0

(a)

	tdma	IP3	IP4	IP5	total
IP0	tdma 0	0	T/2	T/2	T
	tdma 1	T/8	0	3T/4	7T/8
	max	T/8	T/2	3T/4	T
IP1	tdma 0	T/8	0	T/4	3T/8
	max	T/8	0	T/4	3T/8
IP2	tdma 0	T/2	T/2	0	T
	tdma 1	3T/4	0	0	3T/4
	max	3T/4	T/2	0	T
total max		T	T	T	3T / 19T/8

(b)

Figure 3.4: Un ACG (a) et une configuration de flot reconfigurable (b).

d'injection de trafic reconfigurable » (RTIP*). Il est défini par les conditions suivantes :

$$maximiser \sum_{\substack{i,j\in\{1,\dots,P\}\\ k\in\{1,\dots,\Theta_i\}}} c_{ijk}x_{ijk} \tag{3.1}$$

sous les contraintes

$$\sum_{i\in\{1,\dots,P\}} \max_k (c_{ijk}x_{ijk}) \leq T, \qquad \forall j \in \{1,\dots,P\} \tag{3.2}$$

$$\sum_{j\in\{1,\dots,P\}} c_{ijk}x_{ijk} \leq T, \qquad \forall i \in \{1,\dots,P\},\ k \in \{1,\dots,\Theta_i\} \tag{3.3}$$

$$c_{ijk}x_{ijk} \geq c_{ijk}s, \qquad \forall i,j \in \{1,\dots,P\},\ k \in \{1,\dots,\Theta_i\} \tag{3.4}$$

$$x_{ijk} \in \{0,\dots,T\} \tag{3.5}$$

Construire une instance de flot valide consiste à générer une solution optimale ou sous-optimale de ce problème. Un exemple de matrice de flot reconfigurable valide est donné à la figure 3.4-b. Il correspond à l'ACG de la figure 3.4-a. Les équations (3.2) et (3.3) traduisent respectivement les contraintes sur les colonnes et les lignes et donc sur les décompositions de bande passante en réception et en émission respectivement.

Nous pouvons étendre la mesure du niveau de saturation TSL en entrée du réseau,

donnée pour le cas standard, au cas reconfigurable par :

$$TSL = \left(\left(\sum_i \max_k \sum_j c_{ijk} x_{ijk} \right) / P_E T \right) \times 100, \tag{3.6}$$

où P_E est le nombre d'IP émetteurs. Le TSL traduit le niveau maximum d'injection de trafic à un instant donné. Il correspond à une configuration donnée des TDMA qui maximise la quantité de paquets injectés dans le réseau. Cette valeur ne traduit cependant pas les bénéfices de la reconfiguration. Pour cela, nous proposons une mesure de débit visant à refléter le degré de flexibilité obtenu par la reconfiguration. Nous l'appelons « taux maximum de saturation d'un récepteur » (MRT*). Il est défini par :

$$MRT = \left(\left(\sum_i \sum_j \max_k c_{ijk} x_{ijk} \right) / P_R T \right) \times 100, \tag{3.7}$$

où P_R est le nombre d'IP récepteurs. Cette mesure s'applique relativement aux récepteurs. Elle traduit non pas le taux d'injection maximum de paquets dans le réseau à un instant donné, mais le taux maximum d'absorption de paquets par un récepteur, sachant que chaque récepteur peut être considéré à un instant différent, c'est-à-dire chacun dans une configuration de TDMA qui maximise sa réception, et donc son utilisation. Ce calcul a été effectué pour le cas de la matrice de flot de la figure 3.4-b. Le MRT et le TSL sont donnés dans la cellule du coin inférieur droit de la matrice de flot. La figure 3.5 permet d'illustrer également l'intérêt de ces deux coefficients sur un cas simple. Deux configurations de trafic sont possibles suivant (a) et (b). On constate un TSL de 75%, pour un MRT de 100%. Cela signifie dans cet exemple qu'il existe pour chaque récepteur une configuration de TDMA telle qu'il reçoive exactement un paquet par unité de temps. Un récepteur est dans ce cas utilisé à plein. Si, sans perte de généralité, nous avons $P_E = P_R$, nous avons également $TSL \leq MRT$. Dans le cas de flot standard sans reconfiguration, ces mesures sont confondues. Nous pouvons donner le théorème suivant :

Théorème 2. *Pour tout graphe de communication ACG, et tout flot de communication associé, MRT peut être rendu égal à 100% par ajout de nouvelles configurations d'émission, sans création de nouveaux canaux de communication.*

Preuve Il suffit par ajout de tables TDMA (de lignes) aux émetteurs d'augmenter leur taux d'injection vers les récepteurs (colonnes), ceux-ci étant considérés un par un et séquentiellement, en utilisant une communication déjà ouverte dans l'ACG, de manière à saturer le récepteur avec un total de T paquets, éventuellement en retirant des paquets vers les autres récepteurs.

Autrement dit, il est toujours possible d'ajouter des tables TDMA de manière à ce que chaque récepteur soit, à un moment ou à un autre, pleinement utilisé ($MRT = 100\%$) selon les canaux déjà ouverts. Nous allons voir en revanche qu'en ce qui con-

cerne les émetteurs, les mécanismes de reconfiguration asynchrones sont insuffisants dans certains cas pour garantir leur utilisation à plein.

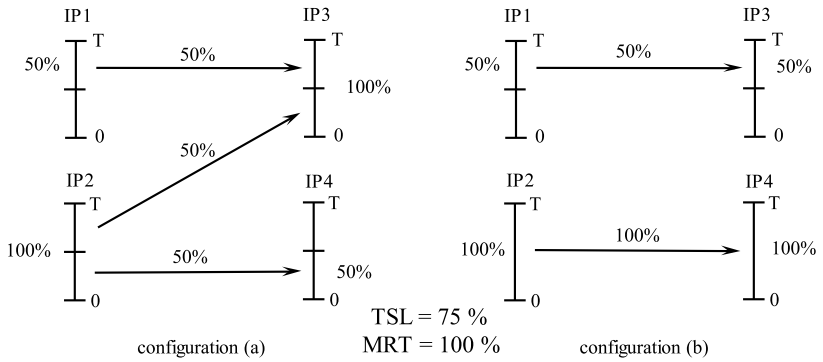


Figure 3.5: Possibilité de saturation maximum d'un récepteur.

3.4 Limites de la reconfiguration asynchrone

Une limite dans la flexibilité obtenue par la technique de reconfiguration asynchrone des émetteurs est illustrée à la figure 3.6. Elle tient à la décomposition de la bande passante en réception qui est fixée une fois pour toute selon tous les canaux ouverts spécifiés dans l'ACG. Si l'on considère les deux configurations (a) et (b) de la figure 3.6, nous pouvons remarquer qu'elles ne peuvent constituer une instance valide du problème de routage reconfigurable CRKPP telle que spécifiée par le RTIP de la section précédente. En effet, l'IP 2 sur la figure ne peut pas modifier sa TDMA, de la configuration (a), à la configuration (b), indépendamment du choix effectué par l'IP 1, sans risquer d'entrer en conflit avec celui-ci. Ainsi, il est impossible d'obtenir une configuration d'utilisation à plein de l'émetteur 2. Ce type de configuration entraîne une violation de la contrainte (3.2) du RTIP portant sur la saturation des colonnes.

D'autres limitations à la flexibilité de l'approche tiennent à la nature des prévisions de trafic réalisées. Les configurations de tables TDMA doivent être spécifiées à l'avance et leur application contrôlée lors de l'exécution par la NI qui doit comporter un système d'évaluation de trafic et de décision. L'ajout de nouvelles configurations a donc un coût non négligeable en terme de matériel, mémoire, surface dans les NI. Ainsi, les configurations ont une utilité sur des périodes d'observation et de fonctionnement relativement longues, lorsque les volumes de données échangés connaissent des modifications substantielles. L'adaptation aux fluctuations rapides et aléatoires du trafic n'est pas complètement prise en compte ici comme avec le trafic BE. Dans la section suivante, nous proposons quelques pistes pour renforcer le caractère adaptatif du NoC GT proposé. Nous considérons des extensions possibles du principe de reconfiguration et le mixage entre trafic GT et BE.

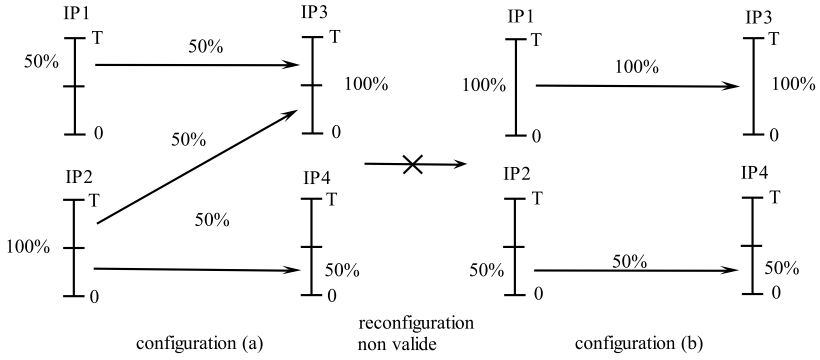


Figure 3.6: Impossibilité de saturation maximum d'un émetteur.

3.5 Extensions des mécanismes de reconfiguration

3.5.1 Niveaux d'adaptation Application/Tâche/Fluctuations

Dans les NoC, l'approche GT est souvent opposée à l'approche BE [23]. Ici, nous proposons d'étudier leurs propriétés de complémentarité pour le traitement adaptatif du trafic. Pour cela, en se basant sur les remarques des sections précédentes, nous distinguons trois niveaux de besoins dans l'adaptation du réseau au trafic. Nous les appelons niveaux Fluctuations, Tâche et Application. Ils correspondent à des périodes d'observation plus ou moins longues du besoin en trafic. Le schéma de la figure 3.7 permet d'illustrer les distinctions faites. Des variations du trafic au cours du temps, pour deux flots de communication dans le réseau y sont représentées. Elles peuvent correspondre à des scénarii différents de la demande en bande passante au cours du temps.

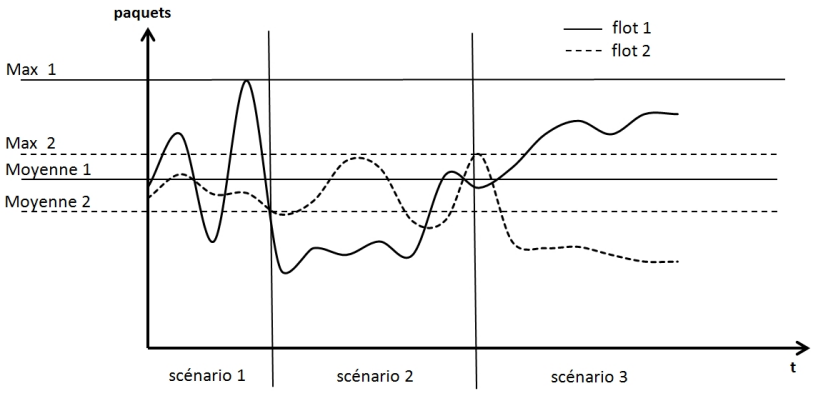


Figure 3.7: Variation de la demande de trafic au cours du temps.

Niveau Fluctuations

Le niveau Fluctuations correspond aux variations rapides de trafic qui n'ont qu'un impact éphémère sur la saturation des canaux. Si le réseau est dimensionné pour du trafic BE évalué en moyenne, les contentions générées sont assez brèves et occasionnelles. Dans le cas du trafic GT, les bandes passantes sont dimensionnées au pire cas et les fluctuations sont sans impact au niveau contention du fait des sur-réservations de *time-slots*.

Niveau Tâche

Le niveau Tâche correspond à des périodes d'observation plus longues que l'on peut considérer comme différents scénarii de la demande en bande passante par les IP. Ici, nous considérons des modifications durables de trafic pouvant être traitées de manière locale par les émetteurs ainsi que dans notre approche de reconfiguration asynchrone des TDMA modélisée par le CRKPP. Cette approche peut s'appliquer pour des cas du type de la figure 3.5. Il faut noter que le principe de la reconfiguration dynamique que nous appliquons pour du trafic GT est transposable pour du trafic BE. Même dans le cas BE, il convient d'éviter une contention structurelle et durable. Généralement, l'allocation des chemins BE tient compte de la demande en trafic (en moyenne) et de la capacité maximum des liens physiques, et cela se ramène à la résolution d'un UFP NP-difficile. En quelque sorte, le principe de la reconfiguration de chemins à la source n'est pas incompatible avec une QoS de type BE, pour éviter des périodes de contention durable dans le réseau.

Niveau Application

Le niveau Application dans notre approche de la reconfiguration correspond également à des scénarii ou modifications durables de la demande en trafic mettant en jeux, non pas un seul émetteur de manière locale, mais plusieurs émetteurs simultanément ainsi que dans le cas exposé à la figure 3.6. Dans un tel cas, pour éviter toute contention possible, le changement des tables TDMA d'émissions ne peut se faire que de manière synchrone entre les différents émetteurs. De tels mécanismes ne sont pas actuellement prévus dans la méthodologie du NoC μSpider II [9], seule la reconfiguration locale est possible. Permettre des changements synchronisés des tables TDMA par les IP demanderait des modifications de l'architecture par ajout soit de compteurs, de lignes de synchronisation, ou de messages de contrôle. Le problème de calcul des chemins non-conflictuels dans le cas de permutations synchrones de tables TDMA, peut être défini comme une extension du problème d'optimisation combinatoire présenté dans ce chapitre pour le cas asynchrone.

Il est admis que l'approche GT peut entraîner un surdimensionnement des ressources. Nous avons proposé d'atténuer ce surdimensionement possible par la reconfiguration dynamique des chemins de communication selon la méthodologie du NoC μSpider II

[9]. Un pas supplémentaire vers plus de souplesse dans l'utilisation des ressources est d'envisager la possibilité d'un couplage entre trafic BE et trafic GT. Nous allons examiner cette possibilité dans le cas du NoC μSpider II qui propose justement des mécanismes de gestion de flux par effet domino en cas de contention.

3.5.2 Routage combiné GT et BE

Nous examinons ici la possibilité d'utilisation des mécanismes de contrôle de flux à effet domino prévus dans la méthodologie du NoC μSpider II [9] et pas utilisés dans l'approche GT présentée dans ce chapitre. Il s'agit de permettre la possibilité de conflits entre paquets et d'étudier la mise en œuvre d'une approche mixte de communication dans le NoC μSpider II avec du trafic BE couplé à du trafic GT.

Il est admis que l'approche BE peut fournir de bonnes performances moyennes avec un réseau non-surdimensionné. Cependant, les périodes de contention peuvent être imprévisibles ou non-prévues, et entraîner une incertitude sur la latence (durée de transfert) des communications. Le trafic GT spécifie cette latence avec certitude au prix d'une pré-réservation exclusive de bande passante. La possibilité d'utiliser conjointement des messages de type BE et de type GT peut être envisagée. Dans [23], le routage conjoint GT et BE est obtenu par un mixage de la commutation de paquets et de circuit. Ici, nous nous plaçons dans le cadre du NoC μSpider II et examinons la prise en compte de messages BE et GT avec la commutation de paquets seule.

Supposons que nous voulons gérer à la fois des messages de type BE et de type GT avec le NoC μSpider II. Un message de type BE est maintenant spécifié par une bande passante moyenne. Cela entraîne que les contraintes temporelles d'émission gérées par les NI doivent être relâchées pour ce type de message. Les tailles de messages BE, ou leur fréquence d'émission, doivent pouvoir varier au delà de la bande passante pré-réservée. Un message GT, en revanche, continue d'être géré selon des contraintes de taille maximum et de date d'émission fixées.

La question est alors de calculer le routage des communications BE et GT pour assurer la qualité de service voulue. Une première sophistication de l'architecture est le mécanisme de gestion des conflits lui-même. Il implique (au minimum) la présence de buffers de mémorisation dans les routeurs, et cela pour chacune de leurs entrées, ce qui n'était pas le cas avec du trafic GT (*contention-free*). Une deuxième difficulté majeure impliquée par la contention est la possibilité de *deadlock* qu'il faut éliminer. L'algorithme de construction de chemins devra éviter les *deadlocks*, ce qui peut impliquer des restrictions dans les possibilités de routage et/ou des coûts supplémentaires s'il faut ajouter des nouveaux canaux ou des nouveaux routeurs.

En supposant une construction des chemins de routage sans *deadlock*, il convient en outre d'évaluer la latence maximum et de garantir celle-ci dans le cas des messages GT. Comme illustré par la figure 3.8, la combinaison du routage GT et BE peut se présenter selon deux perspectives telles que celles qui sont présentées en (a) et (b). En (a), des paquets GT sont introduits sur des liens avec du trafic BE. Cela peut se produire au

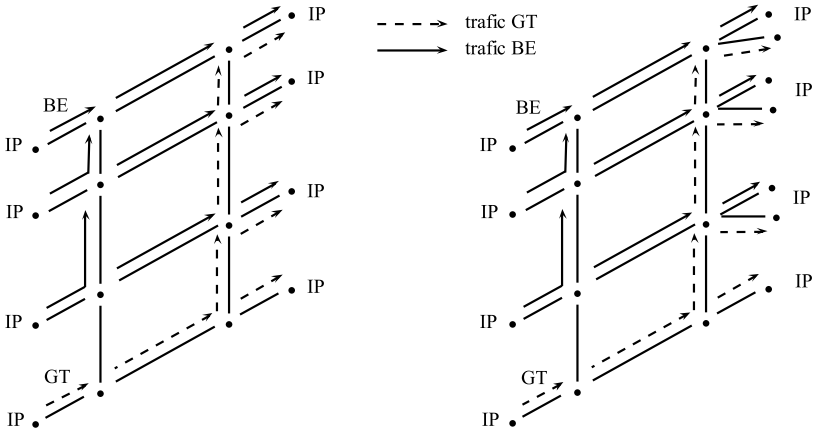


Figure 3.8: Problème de routage conjoint GT et BE. (a) Liens partagés GT BE. (b) Liens dissociés GT BE.

minimum sur le lien terminal vers l'IP destination tel que sur la figure (a). Il s'ensuit que selon la taille des messages en transit dans le réseau, et la possibilité de conflits qui en résulte, des retards (imprévisibles) dans l'acheminement des données de type GT sont à redouter. Un message GT à destination d'un IP éloigné subira des retards au gré des contentions en aval sur son propre chemin, engendrées par l'occupation des liens de type *wormhole*, même si son propre chemin ne comporte que des paquets GT mis à part à l'arrivée. En (b), les deux types de trafic sont intégralement dissociés en ajoutant des liens spécifiques terminaux, et empruntent des chemins de routage totalement disjoints. Le réseau GT n'est ainsi pas perturbé et reste sans contention, tout en utilisant des routeurs communs au réseau BE. Dès lors que des retards sont introduits, garantir une latence maximum des communications GT est un problème complexe et ouvert. Il conviendra dans tous les cas de minimiser le nombre de canaux partagés entre paquets BE et GT et de réduire les tailles de message.

3.6 Conclusion

L'extension de l'approche standard du routage GT au cas reconfigurable a abouti au CRKPP, qui est la formulation d'un problème de routage cyclique, multi-chemin, multi-paquet, d'optimisation combinatoire avec contraintes temporelles. L'approche a pour particularité de permettre, à chaque début de cycle, la modification du choix de la table TDMA d'émission et donc des chemins de données source/destination. L'approche permet d'atténuer le surdimensionnement dû à l'allocation statique des chemins de l'approche GT standard.

Nous avons ensuite présenté et formaliser le problème de la génération de flots de communication valides dans le cas reconfigurable. Ces flots constituent la donnée d'entrée du CRKPP. Cela a permis de mettre en évidence les difficultés inhérentes à la méthode. Malgré la variation dynamique de la bande passante obtenue par la modification asynchrone des tables TDMA à la source, certains cas souhaités de reconfiguration ne peuvent pas être obtenus. C'est le cas particulièrement lorsque la reconfiguration implique plusieurs émetteurs simultanément, ce qui suppose un changement synchrone des tables TDMA des émetteurs.

Pour obtenir plus de souplesse dans l'utilisation des ressources du réseau lors des communications, nous avons examiné la possibilité d'un couplage du routage GT avec du routage BE dans le NoC μSpider II. Dans tous les cas de figure, le coût en ressources devra inclure l'ajout de buffers de mémorisation dans les routeurs qui sont nécessaires au contrôle de flux. Le problème d'optimisation combinatoire que nous traitons dans cette thèse concerne le routage GT uniquement. On peut envisager son extension pour combiner le routage BE et le routage GT. Cela entraîne au minimum de garantir une construction des chemins sans *deadlock* et de minimiser ou éliminer les liens partagés entre chemins BE et GT. Cela constitue une perspective d'extension de nos travaux.

4

Principe des méthodes de résolution des problèmes à garantie de trafic

Les problèmes de routage à garantie de trafic CKPP et CRKPP sont des extensions de problèmes de plus courts chemins multiples avec contraintes combinés avec un *Bin Packing*. Ils sont NP-durs au sens fort. Des tentatives de résolution du premier problème sous forme d'un programme linéaire en nombres entiers, que nous avons menées avec des solveurs standards, ont mis en évidence la difficulté de le résoudre de manière exacte. Les temps d'exécution deviennent rapidement prohibitifs avec l'augmentation de la taille de l'instance. Il apparaît donc justifié de tenter de résoudre ces deux problèmes par des méthodes heuristiques et métaheuristiques. Dans ce cas, nous ne garantissons pas l'obtention d'un optimum global, mais nous recherchons des solutions admissibles et de bonne qualité en temps raisonnable. Pour cela, les résultats devront être validés sur des jeux de tests représentatifs du problème concret. Nous abordons dans ce chapitre la présentation des principes de base des méthodes de résolution heuristiques proposées. Nous proposons comme socle commun aux approches de résolution, l'utilisation d'un graphe spatio-temporel étendu. Celui-ci sert à mémoriser des informations d'occupation temporelle des arcs du réseau, aussi bien dans le cas standard que dans le cas reconfigurable. Son principe est présenté dans la section 4.1. L'ensemble des opérateurs de base de la résolution s'appuient sur cette structure. Dans la section 4.2, nous énonçons une condition nécessaire et suffisante sur l'occupation d'un arc garantissant l'absence de conflit entre paquets et permettant la réutilisation des slots de temps laissés vacants, notamment lors des reconfiguations dynamiques du deuxième problème, c'est-à-dire du CRKPP. Excepté la prise en compte de cette condition spécifique dans le cas reconfigurable, les méthodes de recherche proposées sont strictement identiques pour les deux problèmes. Ces méthodes sont respectivement des recherches locales, un algorithme évolutionnaire et un algorithme mémétique, combinant les deux précédentes, c'est-à-dire une recherche locale et un algorithme évolutionnaire. Leurs principes de fonctionnement sont présentés dans leurs grandes lignes dans la section 4.3. Leurs pseudo-codes sont précisés dans les chapitres 5 et 6 suivants. Une conclusion termine le chapitre.

4.1 Utilisation d'un Graphe Temporel Étendu

Le choix de l'utilisation d'un graphe spatio-temporel étendu (TEG) pour résoudre le CKPP et le CRKPP trouve son origine dans le besoin de mémorisation de l'occupation temporelle des arcs lors de la résolution. Cette mémoire spécifie pour chaque arc du réseau, et pour chaque slot de temps de l'intervalle $[0, T-1]$, si oui ou non l'arc est occupé par un paquet donné au slot de temps considéré. Nous disons qu'un arc est soit « libre » soit « occupé » à l'instant t modulo T. Par conséquent, on associe à chaque arc un tableau de booléens de taille T. Le graphe spatio-temporel ainsi obtenu, est généralement appelé TEG. La notion de TEG a été présentée par Ford et Fulkerson [19] pour modéliser le flot s'écoulant au cours du temps. La taille mémoire nécessaire est en $\mathcal{O}(N^2 \times T)$ pour un graphe général, ou $\mathcal{O}(N \times T)$ pour un graphe planaire, avec N le nombre de nœuds.

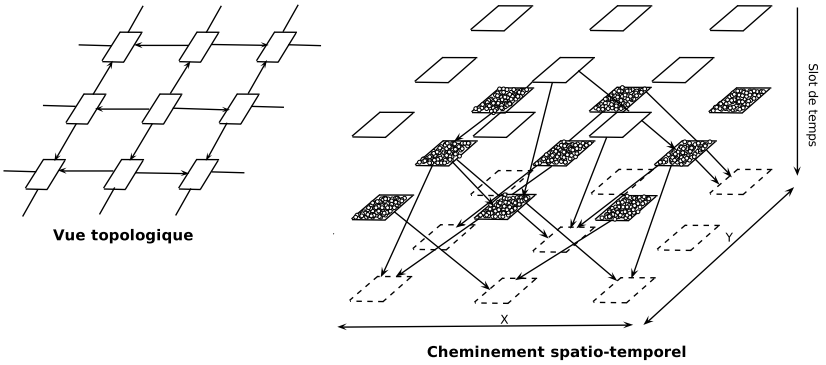


Figure 4.1: Graphe temporel étendu.

Le principe du graphe temporel étendu est illustré par la figure 4.1. Dans sa définition standard, le TEG comporte une copie de l'ensemble des nœuds et des arcs du graphe d'origine à chaque pas discret de temps. Lorsque le fonctionnement est cyclique, comme dans notre cas, chaque strate t est reliée à la prochaine strate $(t+1)$ modulo T. Comme examiné par [34], le concept de TEG permet de résoudre une variété de problèmes de flot s'écoulant dans le temps en appliquant des techniques algorithmiques développés pour des flots dans des réseaux statiques. Dans notre cas, les procédures de construction de chemins sont basées sur l'utilisation d'un TEG. Nous proposons une procédure de construction parallèle gloutonne des chemins qui nécessite une mémorisation de l'occupation temporelle des arcs, et un algorithme de type Dijkstra étendu pour le cas mono-chemin qui utilise un TEG complet, comportant aussi bien la duplication temporelle des arcs que des nœuds. Nous allons voir que cette procédure garantit l'obtention d'un plus court chemin optimal en temps polynomial avec la taille du TEG, en particulier lorsque le message qui s'y rapporte n'est composé que d'un seul paquet. Bien que la taille mémoire dépende du cycle de temps T, cette approche à base

de TEG est adéquate pour nos cas d'application.

4.2 Condition d'accès exclusif à un arc dans le cas reconfigurable

Le TEG est une mémoire qui indique l'état d'occupation d'un arc à chaque slot de temps t modulo T. Dans le cas de routage standard modélisé par le CKPP, une condition nécessaire et suffisante pour garantir une traversée sans conflit d'un arc est que l'arc ne soit pas déjà occupé par un paquet au slot de temps donné t modulo T. Une structure booléenne est suffisante pour mémoriser l'état d'occupation d'un arc. Dans le cas du routage reconfigurable modélisé par le CRKPP, la situation est différente. Une ou plusieurs tables TDMA peuvent être associées à une même NI source. Ces tables peuvent être interchangées à chaque début de cycle d'émission. Ainsi, étant donné un NoC avec un ensemble de chemins sans conflit déjà construits, une question importante est de déterminer à quelle condition un paquet peut transiter par un arc sans conflit et tout en réutilisant les slots de temps laissés vacants en raison d'un changement de table TDMA. Pour ce faire, nous énonçons une condition nécessaire et suffisante qui permet de caractériser la vacance des slots de temps dans la structure du TEG. La résolution du problème dans le cas reconfigurable dépend essentiellement de la prise en compte de cette condition.

Pour énoncer cette condition, nous avons besoin de la notion de « temps relatif » de franchissement d'un arc. Cette valeur correspond à l'instant de franchissement de l'arc compté relativement à un début de cycle d'émission donné. Étant donné l'occurrence du j^{eme} paquet d'un message μ, émis à l'instant $t(\mu) \in [0, T-1]$, lors du cycle kT, $k \geq 0$, et qui suit le chemin $v_0, v_1, \ldots, v_n$, nous disons que le paquet traverse l'arc (v_i, v_{i+1}) au temps relatif $t_r = t(\mu)+i+j$. En nous basant sur une telle valeur relative au cycle d'émission courant, nous pouvons maintenant énoncer une condition nécessaire et suffisante qui assure une traversée sans conflit d'un arc. Cette condition est énoncée par le théorème suivant:

Théorème 3. *Étant donné un NoC comportant un ensemble de chemins construits sans conflit, un paquet peut franchir un arc donné sans conflit au temps relatif t_r si et seulement si*

1. *l'arc est libre à l'instant t_r modulo T, ou*
2. *l'arc est occupé à l'instant t_r modulo T par un paquet issu d'une autre table TDMA liée à la même NI source et tel que $t_r = t'_r$, avec t'_r le temps relatif du franchissement de l'arc par ce paquet.*

Preuve Si un conflit existe entre deux paquets à un instant donné, nous avons nécessairement $kT + t_r = k'T + t'_r$, pour certains entiers k, k'. Supposons que $t_r \neq t'_r$,

la seule possibilité d'avoir un conflit, est que $k \neq k'$, ce qui signifie que les paquets ont été émis à des cycles différents. Cette possibilité de conflit doit être impérativement évitée. Le paquet ne peut donc pas traverser l'arc sans conflit. Inversement, supposons que $t_r = t'_r$. S'il y a un conflit, il s'ensuit que $k = k'$, ce qui signifie que les paquets ont été émis lors d'un même cycle. Mais, étant donné que les paquets provenant de deux TDMA différentes d'une même NI sont nécessairement émis de façon mutuellement exclusive à chaque cycle, il ne peut pas y avoir conflit. Si $t_r = t'_r$, le paquet peut traverser l'arc sans conflit. Ceci termine la preuve.

Cette condition garantit qu'aucun conflit ne peut se produire, et que tous les slots de temps « libérés » , c'est-à-dire qui ne sont pas occupés en raison d'un échange de table TDMA, peuvent être réoccupés par des paquets de la nouvelle table TDMA émise. Les nouveaux messages peuvent donc réutiliser l'intégralité des slots de temps vacants. Pour mettre en œuvre l'évaluation de cette condition dans la structure du TEG, de nouveaux éléments de mémorisation doivent être ajoutés à chaque arc du TEG, à chaque slot de temps. Ce sont, l'identifiant de la NI source, les identifiants des tables TDMA de cette NI auxquelles appartiennent les paquets, et l'instant relatif de passage correspondant. Tandis que la résolution du CKPP nécessitait juste un booléen par arc et par slot de temps, la taille de la nouvelle structure indiquant le statut d'un arc pour le CRKPP est en $\mathcal{O}\left(T \log\left(N \times T\right)\right)$, au lieu de $\mathcal{O}\left(T\right)$ pour le CKPP. L'avantage est la possibilité de réutilisation complète des slots de temps vacants accessibles d'une manière mutuellement exclusive.

4.3 Principe des méthodes de recherche

Pour résoudre le problème d'optimisation combinatoire par le moyen de procédures heuristiques, nous avons opté pour trois types de méthodes de recherche dont nous présentons ici le principe. Ces méthodes sont toutes basées sur l'utilisation d'un TEG et d'un ensemble d'opérateurs de base simples communs aux méthodes. Ce sont respectivement des recherches locales itérées, un algorithme évolutionnaire, et un algorithme mémétique combinant la recherche locale au sein d'un algorithme évolutionnaire. Les opérateurs de base ainsi que les pseudo-codes des méthodes de recherche seront présentés dans les chapitres 5 et 6 suivants. Il convient de noter que la différence entre la résolution des cas standard et reconfigurable du problème tient uniquement à l'évaluation de la condition d'accès à un arc spécifiée précédemment dans le théorème 3. Les méthodes de recherche restent identiques à la fois pour le CKPP et le CRKPP.

Dans cette section, nous ne donnons que le principe général des méthodes sans entrer dans le détail des algorithmes. Ces principes déterminent différents modes de recherche et d'exploration de l'espace des solutions. Dans tous les cas de figure, une ou plusieurs solutions sont modifiées de manière itérative selon un opérateur de voisinage qui consiste à détruire puis reconstruire un nombre variable de chemin d'une solution courante. L'opérateur est stochastique en ce que le choix de la modification locale est

en partie aléatoire.

4.3.1 Principe des recherches locales itérées

La recherche locale consiste en une modification locale d'une solution courante en considérant un voisinage plus ou moins large de cette solution. Cette solution courante, pas nécessairement admissible, est obtenue au départ par une méthode de construction rapide. Les modifications locales déterminent une intensification de la recherche dans une zone plus ou moins restreinte de l'espace des solutions. La réitération de la recherche locale à partir de solutions initiales aléatoires permet de diversifier la recherche [32].

En se basant sur une solution de départ préalablement construite mais pas nécessairement admissible, trois versions de la stratégie de recherche locale sont considérées. La première version est une méthode de recherche aléatoire itérée, de type marche aléatoire, que nous appelons (IRS*). La deuxième est une méthode de recherche locale gloutonne de type premier-meilleur que nous appelons (ILS-FI*). La troisième est une recherche locale en profondeur que nous appelons (ILS BI*).

La méthode IRS fait évoluer une solution courante en effectuant un nombre donné de mouvements successifs dans le voisinage, puis sélectionne la meilleure solution rencontrée. Chaque solution examinée est obtenue par modification de la solution précédente à l'aide de l'opérateur de voisinage. Une suite de modifications de la solution courante est exécutée, leur nombre étant fixé. La meilleure solution rencontrée est sélectionnée. Le procédé complet est réitéré un certain nombre de fois après permutation des dates d'émission et réinitialisation des chemins. La méthode est très simple puisque très peu d'opérations de copie de données sont effectuées à chaque d'itération.

Dans les recherches locales ILS-FI et ILS-BI, une solution donnée constitue l'élément pivot autour duquel est effectuée la recherche dans le voisinage. Chaque nouvelle solution examinée est obtenue par modification de la solution pivot avec l'opérateur de voisinage. Les deux versions diffèrent par la règle de pivotage choisie. Dans ILS-FI, la première solution rencontrée de qualité supérieure à la solution pivot devient le nouveau pivot.

Dans la version ILS-BI, la meilleure solution rencontrée à l'intérieur du voisinage est sélectionnée comme le nouveau élément pivot. Seul un échantillon de taille limitée du voisinage est examiné.

Dans les deux versions ILS-FI et ILS-BI, la recherche locale est stoppée lorsqu'aucune amélioration n'est trouvée, c'est-à-dire lorsqu'on a atteint un minimum local. En pratique, seul un échantillon du voisinage est examiné car il s'agit ici d'un voisinage large et d'un opérateur stochastique. Dans les trois méthodes précédentes, la taille du voisinage est déterminée par le nombre de chemins supprimés et reconstruits par l'opérateur du même nom, à chaque pas d'exécution. La taille du voisinage ne doit pas être confondue avec la taille de l'échantillon examiné, c'est-à-dire le nombre maximum de solutions examinées obtenues à partir d'un élément pivot. Ensuite, le procédé complet est réitéré

un certain nombre de fois après permutation des dates d'émission et réinitialisation des chemins de même que dans la version IRS.

4.3.2 Principe de l'algorithme évolutionnaire

Nous proposons une autre manière d'explorer l'espace des solutions suivant le paradigme des algorithmes évolutionnaires [62]. La recherche procède maintenant par une application simultanée des opérateurs de base, de voisinage et construction, à un ensemble (une population) de solutions. Suivant la terminologie des algorithmes évolutionnaires, les solutions sont maintenant des « individus » qui doivent répondre aux exigences du problème. Pour évaluer la qualité de la solution, une fonction appelée *fitness* associe une valeur scalaire à chaque individu de la population. Les « meilleurs » individus ont la plus grande fitness, tandis que les « mauvais » individus une fitness plus faible. Des opérateurs de sélection sont ajoutés. Une sélection permet le remplacement des « mauvaises » solutions de la population par les « meilleures » . Un opérateur de mutation participe à la diversification. Il est mis en œuvre par des permutations et modifications des dates d' envoi de messages dans le réseau et reconstruction de chemin. Un deuxième opérateur de mutation, reprend le principe de la modification locale d'une solution et permet l'intensification de la recherche. Il s'agit de l'opérateur de voisinage utilisé dans les recherches locales de la section précédente qui consiste en la destruction et reconstruction d'un sous-ensemble de chemins. Il ne s'agit pas à proprement parler d'un algorithme génétique car ici le codage de la solution est direct et les opérateurs appliqués sont spécifiques au problème. Par l'approche évolutionnaire, une construction aussi rapide que possible et diversifiée de solutions admissibles est recherchée. La figure 4.2 vise à illustrer le principe de la recherche en parallèle effectuée par l'approche évolutionnaire au sein de l'espace des solutions.

4.3.3 Principe de l'algorithme mémétique

Si nous introduisons au sein de l'algorithme évolutionnaire de la section 4.3.2, le principe des recherches locale itérées de la section 4.3.1, nous obtenons un algorithme évolutionnaire hybride. Ce type d'algorithme à base de population et dans lequel des recherches locales sont appliquées en tant qu'opérateur de mutation est généralement référencé dans la littérature sous la dénomination d'algorithme mémétique [49]. La figure 4.3 nous permet de schématiser le principe de recherche de solution par l'algorithme mémétique. Sur la figure sont illustrées les recherches locales exécutées en parallèle et conduisant à un minimum local. La multiplication de ces recherches locales couplée aux opérateurs de sélection et de mutation détermine la dynamique de recherche. L'objectif est de réutiliser les avantages respectifs des méthodes précédentes dans une seule et même méthode de recherche.

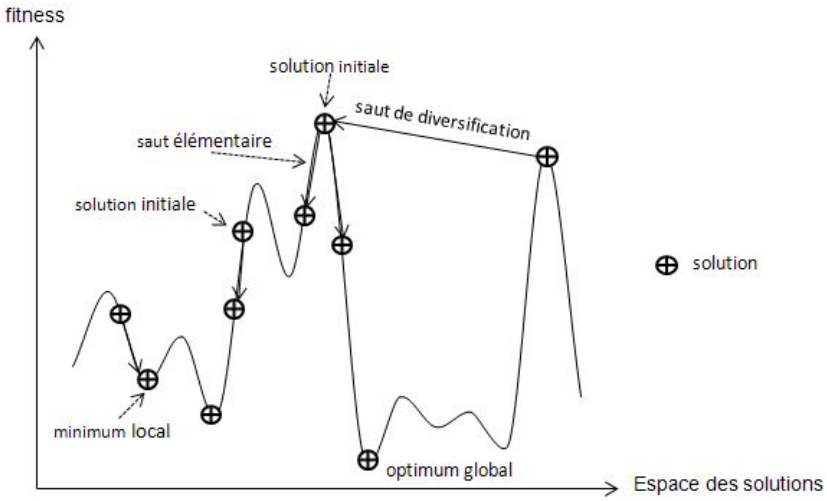


Figure 4.2: Principe de l'algorithme évolutionnaire.

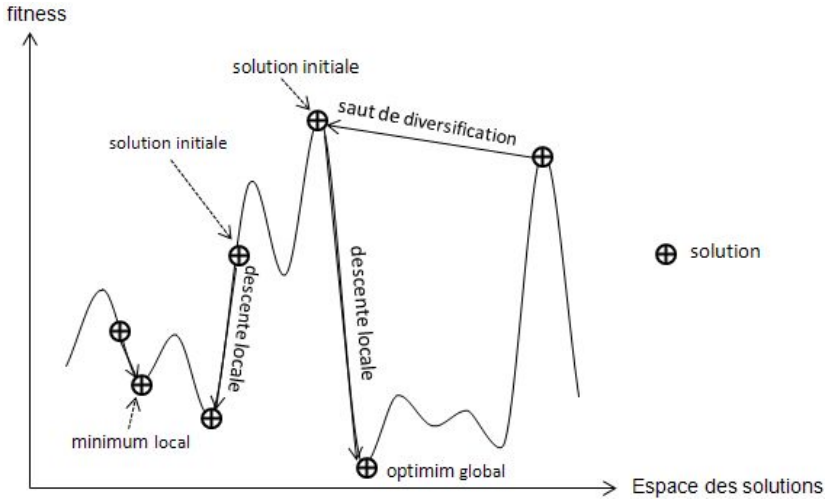


Figure 4.3: Principe de l'algorithme mémétique.

4.4 Conclusion

Les principes généraux appliqués pour la résolution des problèmes de routage ont été présentés. La prise en compte des contraintes temporelles du CKPP et du CRKPP amènent naturellement à l'utilisation d'un graphe spatio-temporel étendu, appelé TEG, qui permet une mémorisation de l'occupation temporelle des arcs du réseau. Concernant le CRKPP, une condition nécessaire et suffisante portant sur l'occupation des arcs a été explicitée pour, à la fois éviter l'apparition de conflits entre paquets et permettre la réutilisation des slots de temps laissés vacants lors des reconfigurations dynamiques de chemins. Les méthodes de résolution proposées pour les problèmes standard et reconfigurable sont toutes basées sur l'utilisation d'un TEG et la vérification de cette condition d'occupation. Les principes des méthodes de recherche proposées ont été présentés dans leurs grandes lignes. Nous proposons d'étudier trois types de méthodes. Le premier type correspond à des recherches locales itérées. Le deuxième type de méthode à un algorithme évolutionnaire. Le troisième type à une combinaison des deux précédents sous la forme d'un algorithme mémétique. Il s'agit d'un algorithme évolutionnaire incluant une recherche locale en tant qu'opérateur de variation. Nous allons préciser maintenant le détail des opérateurs communs de base aux méthodes et donner les pseudo-codes des approches proposées.

5
Opérateurs de base

Pour résoudre le problème d'optimisation combinatoire selon le principe des procédures heuristiques, la conception d'un certain nombre d'opérateurs de base pour la manipulation des solutions est fondamentale. Nous les appelons opérateurs de base pour leur implication essentielle dans tous les compartiments des méthodes de recherche dont les principes ont été présentés au chapitre 4. Nous définissons quatre types d'opérateurs. Le premier type d'opérateur sert à manipuler et gérer un ensemble d'instants de départ des messages. Un premier opérateur de gestion effectue des permutations de deux dates de messages dans une même TDMA. Un deuxième opérateur de gestion applique des translations d'une unité de temps sur les dates de départ d'une même TDMA. Un deuxième type d'opérateur consiste à tenter une construction simultanée de l'ensemble des chemins non-conflictuels origine/destination aussi vite que possible, par un procédé de choix glouton, réalisé en parallèle, du prochain sommet à insérer dans le chemin. Lors de la construction, étant donné un chemin en cours de construction, le sommet suivant inséré est le premier sommet trouvé compatible avec le passage des paquets. Les chemins sont construits simultanément, de façon parallèle, pas à pas, arc par arc, chemin par chemin. Un troisième type d'opérateur que nous proposons est une construction séquentielle des chemins par un algorithme de type Dijkstra modifié opérant dans le TEG. Il permet de construire un chemin unique dans un NoC occupé. Il garantit l'obtention d'un chemin optimal en temps polynomial avec la taille du TEG dans le cas d'un message mono-paquet, et reste efficace dans le cas multi-paquet. Le quatrième type d'opérateur de base est un opérateur de voisinage pour réaliser des améliorations locales, ou réparations, dans une solution donnée. Il est conçu pour la recherche locale. De façon générale, la structure du TEG est utilisée par tous les opérateurs de base. En intégrant les opérateurs de base dans des procédures plus élaborées, nous avons spécifié des méthodes de recherche qui seront détaillées dans le chapitre 6 suivant. La mise en œuvre de ces méthodes de recherche est identique pour les deux problèmes CKPP et CRKPP à l'exception du respect des conditions du théorème (3) pour le problème avec reconfiguration dynamique CRKPP. Le corps de ce chapitre est consacré à la présentation des quatre opérateurs de base via leur pseudo-code. Les deux opérateurs de gestion des dates d'émissions sont présentés dans la section 5.1. L'opérateur de construction parallèle gloutonne dans la section 5.2. L'algorithme

mono-chemin de type Dijkstra étendu au TEG est adapté aux aspects cyclique et multi-paquet du problème est présenté dans la section 5.3. L'opérateur de réparation et d'amélioration, identifié à un opérateur de voisinage, est présenté et détaillé dans la section 5.4. Une conclusion met fin au chapitre.

5.1 Gestion des dates d'émission

Ainsi que nous l'avons vu dans la définition des deux problèmes, l'admissibilité des chemins construits entre des nœuds source/destination dépend des dates d'émission des messages qui sont spécifiées dans l'intervalle $[0, T-1]$ de chaque table TDMA. Ces dates constituent des variables essentielles du problème qu'il convient de faire évoluer lors des réitérations des constructions de chemins. Deux opérateurs sont proposés pour gérer les dates d'émission avant chaque construction. Ils sont décrits par les Algorithmes 1 et 2. Une table TDMA donnée étant choisie au hasard, ils consistent respectivement à effectuer soit une permutation aléatoire entre deux dates d'émission, soit une translation d'une unité de temps des dates d'émission de cette TDMA. Les dates de départ sont ajustées en tenant compte du nombre de paquets des messages.

Algorithme 1: permuteMessages

Entrées : $S \in$ Solution
Sorties : S
begin
  Choisir aléatoirement une TDMA δ;
  Choisir aléatoirement deux messages $i, j \in \delta$;
  // Permuter leurs dates d'émission $tdep_i, tdep_j$
  $S \leftarrow permute(S, \delta, i, j)$;
  retourner S;

Algorithme 2: translateDepartureDates

Entrées : $S \in$ Solution
Sorties : S
début
  Choisir aléatoirement une TDMA δ;
  pour *chaque message* $i \in \delta$ **faire**
    $tdep_i \leftarrow (tdep_i + 1) \bmod T$;
  retourner S;

5.2 Construction parallèle gloutonne

Cette procédure construit les chemins source/destination pas à pas et de façon parallèle et gloutonne. Afin que cette procédure puisse fonctionner, un ensemble de tables de routage sont supposées construites préalablement. Elles indiquent pour chaque nœud du graphe sa distance la plus courte à chaque destination possible. Ces tables sont construites une fois pour toute lors de l'initialisation globale de l'algorithme à partir du graphe de communication spécifiant la topologie du NoC. Le pseudo-code de l'heuristique de construction gloutonne est donné par l'Algorithme 3. Lors de l'exécution de cet opérateur, les chemins origine/destination sont construits pas à pas et de façon conjointe. Chaque pas d'itération correspond à un slot de temps donné, et augmente la taille des chemins d'une unité. Le routeur suivant est choisi en fonction de la plus petite distance au nœud de destination et selon que la contrainte de disponibilité de l'arc choisi est vérifiée pour tous les paquets du message. Cette contrainte est vérifiée à l'aide des informations contenues dans le TEG. Le statut de l'arc est ensuite mis à jour dans le TEG suivant le choix effectué. Si plusieurs choix sont possibles, celui-ci est effectué de manière aléatoire. Le processus se poursuit ainsi message après message pour chaque slot de temps. Le processus s'arrête lorsque tous les messages ont atteint leur destination avec succès, ou sont bloqués en l'absence d'un routeur successeur valide. La solution obtenue peut être admissible ou non.

5.3 Dijkstra modifié dans le TEG

Nous présentons un algorithme de type Dijkstra modifié permettant de construire un plus court chemin unique source/destination dans le graphe spatio-temporel étendu. Ici, le TEG contenant indifféremment des arcs « libres » ou « occupés » est considéré comme la donnée d'entrée de l'algorithme. L'algorithme classique de type Dijkstra étendu au TEG est adapté au cas particulier où les coûts des arcs sont constants et valent une unité de temps, tout en tenant compte de la nature cyclique du problème. Nous verrons que l'algorithme garantit l'optimalité dans le cas d'un message à paquet unique (avec ou sans émission cyclique), alors que cela n'est pas le cas pour des messages multi-paquets. A chaque nœud du réseau (le graphe du NoC initial) est associé un tableau de marquage de taille T qui spécifie si ce nœud a déjà été visité lors de la construction du chemin au slot de temps t modulo T, dans ce cas nous disons que le nœud est « scanné » au slot de temps t modulo T. Ce tableau précise également pour chaque nœud et slot de temps son prédécesseur dans le chemin en cours de construction. La procédure est donnée dans l'Algorithme 4. Dans un algorithme à marquage d'étiquette (*label setting algorithm*)[68], comme l'algorithme classique de Dijkstra, le nœud sélectionné pour le marquage est toujours à une distance minimale de l'origine. Aussi, une fois que le nœud est analysé, il est définitivement étiqueté comme scanné et ne sera plus jamais sélectionné. Ici, il y a une simplification par rapport à

Algorithme 3: Construction parallèle gloutonne (constructSolutionPar)

Entrées : $S \in$ Solution
Sorties : S
début
  tant que *tous les messages ne sont pas déclarés « construit » ou « bloqué »* **faire**
    pour *chaque* $ni \in NI$ **faire**
      pour *chaque TDMA* $\delta \in ni$ **faire**
        pour *chaque message* $i \in \delta$ **faire**
          `// Déterminer le dernier nœud du chemin`
          $u \leftarrow path_i\,[length_i - 1]$;
          `// Déterminer le slot de temps correspondant`
          $t_i \leftarrow tdep_i + length_i - 1$;
          Chercher un successeur v de u qui minimise la distance au nœud destination et tel que l'arc (u, v) dans le TEG est « libre » pour les paquets du message aux slots de temps correspondants;
          S'il y a plus d'un successeur candidat, faire un choix aléatoire;
          Si la destination est atteinte, le chemin est déclaré « construit » ;
          S'il n'y a pas de nœud admissible v, le chemin est déclaré « bloqué » ;
          Mettre à jour le statut de l'arc dans le TEG;
  retourner S;

l'algorithme standard de Dijkstra due au coût unitaire associé à chaque arc. Le besoin de sélectionner un nœud à distance minimale dans une liste non ordonnée ne s'impose pas. Chaque itération de la boucle principale correspond à une incrémentation d'un slot de temps, augmentant la longueur du chemin d'une unité. Tous les successeurs d'un nœud scanné sont nécessairement à distance minimale lors de la prochaine itération. Par conséquent, ils constitueront les nœuds scannés à l'itération suivante. Ils sont mémorisés dans le tableau $VisitedNodes_1$ de l'Algorithme 4. Une image du comportement de cet algorithme est celle d'une onde qui se propage à vitesse constante dans toutes les directions, à partir du nœud origine jusqu'à ce qu'elle atteigne ou pas le nœud de destination. En raison des coûts unitaires, un nœud successeur peut avoir à chaque slot de temps, plus d'un prédécesseur possible. Dans ce cas, un seul des prédécesseurs possibles est choisi, comme indiqué dans la boucle interne qui met à jour l'attribut $succ.pred\,[(tstep + 1) \bmod T]$. Le choix se fait aléatoirement. Des appels multiples de la procédure permettent de diversifier les chemins possibles obtenus.

Dans le cas d'un message mono-paquet, l'algorithme fonctionne exactement comme une procédure classique de type Dijkstra, qui garantit l'obtention d'un chemin de lon-

Algorithme 4: Construction de chemin avec Dijkstra modifié

Entrées : $S \in$ Solution, $Mess \in$ Message
Sorties : $Path$
début
  $VisitedNodes_1, VisitedNodes_2$: Ensemble de nœuds;
  $Path$: Chemin;
  $prec, succ$: Nœud;
  $tstep$: Entier;
  Initialiser le tableau de marquage de taille T pour chaque nœud à $faux$;
  $VisitedNodes_2 \leftarrow \{Mess.origin\}$;
  $tstep \leftarrow Mess.tdep$; `// instant de départ`
  tant que $VisitedNodes_2$ *non vide* **et** *destination non atteinte* **faire**
    pour chaque nœud $prec \in VisitedNodes_2$ **faire**
      $prec.scanned\,[tstep \bmod T] \leftarrow vrai$;
    $VisitedNodes_1 \leftarrow VisitedNodes_2$; $VisitedNodes_2 \leftarrow \{\}$;
    pour chaque nœud $prec \in VisitedNodes_1$ **faire**
      Inscrire le statut des arcs d'origine $prec$ du chemin courant dans le TEG;
      pour chaque nœud $succ \in successors(prec)$ **faire**
        si non $succ.scanned\,[(tstep+1) \bmod T]$ **et** *le statut de l'arc* $(prec, succ)$ *dans le TEG est « libre » pour les paquets du message à partir du slot de temps* $tstep \bmod T$ **alors**
          si $succ.pred\,[(tstep+1) \bmod T] = null$ **alors**
            $succ.pred\,[(tstep+1) \bmod T] \leftarrow prec$;
            $VisitedNodes_2 \leftarrow VisitedNodes_2 \cup \{succ\}$;
          sinon
            si $rand(0,1) > 0.5$ **alors**
              $succ.pred\,[(tstep+1) \bmod T] \leftarrow prec$;
          si $succ = Mess.destination$ **alors**
            Fixer la destination comme atteinte;
      Effacer le statut des arcs d'origine $prec$ du chemin courant dans le TEG;
    $tsep \leftarrow tsep + 1$;
  si *destination atteinte* **alors**
    Fixer le chemin comme « construit » ;
    Reconstruire le chemin $Path$ en partant du nœud destination vers le nœud origine;
    Mettre à jour le TEG avec ce nouveau chemin;
  sinon
    Fixer le chemin comme « bloqué » ;
  return $Path$;

gueur minimale dans le TEG. A chaque itération de la boucle principale, et donc à chaque pas de temps, les nœuds examinés sont nécessairement à une distance minimale de l'origine. Un nœud donné ne peut être examiné deux fois à un slot temps t modulo T, et la procédure s'arrête dès lors qu'une destination est atteinte ou lorsque il n'y a plus de nœud à examiner. Le nombre de slots de temps vérifiés à chaque nœud dépend de la taille du message et du degré du graphe. Par conséquent, la complexité temporelle de l'algorithme est en $\mathcal{O}(N \times T^2)$ pour des graphes planaires, et $\mathcal{O}(N^2 \times T^2)$ pour des graphes généraux, avec N le nombre de nœuds. Puisque dans le cas de messages mono-paquets, l'algorithme garantit l'optimalité en temps polynomial avec la taille du TEG, il est pseudo-polynomial relativement à la taille de l'instance du CKPP ou CRKPP. La complexité spatiale est en $\mathcal{O}(N \times T)$ pour des graphes planaires, ce qui correspond à la taille mémoire du TEG. Alors que l'algorithme garantit l'optimalité dans le cas d'un message mono-paquet, cela n'est plus vrai dans le cas d'un message multi-paquet. Dans ce cas, la possibilité de conflits intra-message implique la mise à jour dans le TEG du statut des arcs du chemin en cours de construction à partir du nœud en cours d'examen, plus particulièrement des arcs du chemin ayant ce nœud pour origine à d'autres slots de temps. La mise à jour est effectuée pour chaque nœud scanné au slot de temps courant par les deux instructions « Inscrire le statut des arcs d'origine $prec$ ... » et « Effacer le statut des arcs d'origine $prec$ ... » , respectivement au début et à la fin de la boucle externe « Pour » dans l'Algorithme 4. Puisque que les circuits sont autorisés dans un chemin en cours de construction, des paquets d'un même message peuvent se croiser à un nœud donné, mais ne doivent pas être en conflit sur des arcs sortant du nœud en cours d'examen. Pour un cycle d'émission donné, nous devons nous assurer que le paquet d'entête d'un message ne viendra pas percuter la queue de cette même occurrence de message. Entre différents cycles d'émission, nous devons nous assurer que les paquets d'une occurrence de message ne viendront pas percuter les paquets d'une autre occurrence de ce message émise lors d'un cycle ultérieur. La survenue de telles situations est anticipée dans les deux instructions ci-dessus, par la mise à jour dans le TEG du statut des arcs d'origine $prec$ du chemin en cours de construction. Pour résumer, il faut noter que par examen du TEG, les circuits de longueur kT, $k > 0$, dans le chemin en cours de construction ne peuvent être admis. Un nœud donné ne peut pas être examiné deux fois au même slot de temps t modulo T. En revanche, les circuits de longueur $kT + i$, $k \geq 0$, $1 < i < T$, sont autorisés dans le chemin en cours de construction. Ainsi que nous l'avons vu, pour éviter des conflits intra-messages le TEG est mis à jour pour le nœud en cours d'examen. Il s'ensuit que le choix du nœud successeur possible peut dépendre des choix précédemment effectués pour le chemin, notamment lorsqu'un même nœud est à nouveau examiné lors d'un slot de temps t' modulo T différent. Dans ce cas, les canaux de sortie peuvent être occupés différemment selon les choix antérieurs effectués. Il s'agit d'une indication d'un problème plus complexe. Aussi, l'optimalité du chemin calculé n'est pas garantie dans le cas d'un message multi-paquet. Savoir si oui ou non, le problème de plus court chemin multi-paquet admet un algorithme en temps polynomial avec la taille du TEG

est une question ouverte. Cependant, l'optimalité du chemin trouvé reste garantie si aucun conflit possible intra-message n'a pas été détecté au cours de la construction du plus court chemin. Cela peut être vérifié lors de la construction. Nous constatons aussi que la probabilité d'obtenir un chemin origine/destination optimal, est inversement proportionnelle au nombre de paquets.

5.4 Opérateur de voisinage

Cet opérateur effectue une modification locale de la solution et joue le rôle d'opérateur de voisinage dans les recherches locales et métaheuristiques considérées. L'opérateur supprime aléatoirement un sous-ensemble de chemins « construits » ou « bloqués » d'une solution, puis les reconstruit avec la méthode de construction parallèle gloutonne et l'algorithme de Dijkstra modifié pour le cas mono-chemin. Cet opérateur de voisinage est utilisé en phase d'amélioration de la solution réalisant l'intensification de la recherche dans une zone plus ou moins restreinte de l'espace des solutions. Le pseudo-code est donné dans l'Algorithme 5. Le paramètre $nbMsg$ correspond au nombre de chemins supprimés, il détermine la taille du voisinage. Nous pouvons interpréter cet opérateur comme une procédure de « réparation » suivant le principe « ruiner et récréer » (*ruin and recreate*) ainsi que formulé par [60]. Son application répétée vise à transformer des solutions non-admissibles en solutions admissibles avec un coût réduit en temps d'exécution.

Algorithme 5: generateNeighbor

Entrées : $S \in$ Solution, $nbMsg \in$ Entier
Sorties : S
début
  // Supprimer $nbMsg$ chemins de S et mettre à jour le TEG
  $S \leftarrow removeMessages(S, nbMsg)$;
  $S \leftarrow constructSolutionPar(S)$; // construction parallèle gloutonne
  $S \leftarrow constructSolutionWithModifiedDijkstra(S)$; // construire les chemins restants un à un dans le TEG
  retourner S;

5.5 Conclusion

Les opérateurs de base, que nous avons conçu, pour la construction et la transformation des solutions et communs aux méthodes de recherche ont été présentés. L'opérateur de gestion des dates d'émissions permet d'obtenir une grande diversité des dates d'envoi des message, augmentant ainsi les possibilités de construction des chemins les plus courts. L'opérateur de construction parallèle gloutonne, cherche à construire l'ensemble

des chemins non-conflictuels source/destination aussi vite que possible par un procédé de choix glouton du prochain sommet à insérer dans le chemin. L'opérateur Dijkstra modifié dans le TEG, construit un chemin unique dans un NoC occupé. Il garantit l'obtention d'un chemin optimal en temps polynomial dans le cas mono-paquet, et reste efficace dans le cas multi-paquet. L'opérateur de voisinage, permet de transformer, par son utilisation répétée, des solutions non admissibles en solutions admissibles avec un coût réduit en temps d'exécution. Nous les avons appelés opérateurs de base. Nous avons donné le pseudo-code de ces opérateurs afin d'en donner une compréhension précise et permettre leur reproduction et réutilisation par des tiers. Il convient maintenant de préciser le pseudo-code des méthodes de recherche elles-mêmes.

6

Algorithmes de recherche heuristiques et métaheuristiques

Le développement de méthodes heuristiques est justifié par la difficulté du problème de routage qui est NP-difficile. Nous avons constaté par ailleurs l'impossibilité pratique d'une résolution exacte du problème sur nos instances de grandes tailles, cela en utilisant le solveur (GLPK*) sur notre modèle linéaire en nombres entiers du problème. Nous présentons dans ce chapitre le détail des algorithmes heuristiques utilisés. Les heuristiques considérées sont trois variantes de recherches locales itératives, opérant sur une solution unique, un algorithme évolutionnaire opérant sur une population de solutions et un algorithme mémétique incorporant une recherche locale au sein d'un algorithme évolutionnaire.

La recherche locale fonctionne par des mouvements de faible ampleur dans un voisinage de la solution courante. La fonction de voisinage détermine la solution suivante obtenue à partir d'une modification locale de la solution courante. Lorsqu'aucune amélioration de la solution ne peut être obtenue dans le voisinage, la solution en cours est un optimum local relatif à la fonction de voisinage. La réitération de la recherche locale à partir de conditions de départ aléatoires permet de diversifier les zones d'exploration. Chaque implantation proposée réutilise et combine les quatre opérateurs de base présentés dans le chapitre 5. L'opérateur de voisinage est à la base du principe de la recherche locale. Nous avons retenu trois versions de la recherche locale suivant le mode d'examen du voisinage retenu. La première version est une recherche de type marche aléatoire. Les deux autres versions sont une recherche gloutonne *first improvement search* une recherche en profondeur *best improvement search*. Ensuite, en combinant les mêmes opérateurs de base dans un schéma d'algorithme à base de population, avec des opérateurs de sélection sur les solutions, nous spécifions un algorithme évolutionnaire. Puis, pour tenter de tirer parti des avantages de l'algorithme évolutionnaire et de la recherche locale conjointement, nous avons inclus celle-ci en tant qu'opérateur dans un algorithme évolutionnaire selon le principe d'un algorithme mémétique.

Dans la section 6.1 de ce chapitre, nous présentons les recherches locales itérées sous trois différentes versions. L'algorithme évolutionnaire, opérant sur une population

de solutions, et l'algorithme mémétique, combinant l'algorithme évolutionnaire et la recherche locale sont présentés dans la section 6.2. Une conclusion termine ce chapitre.

6.1 Recherches locales itérées

Les recherches locales combinent les opérateurs de base du chapitre précédent selon différentes stratégies de recherche. Elles opèrent toutes à partir d'une solution unique préalablement construite et non nécessairement admissible. Elles diffèrent par le mode de parcours d'un voisinage autour de la solution courante. Elles sont réitérées avec des dates d'émission de messages modifiées.

6.1.1 Boucle itérative externe commune aux recherches locales

La recherche locale itérée [32] est l'une des plus simples méthodes de recherche heuristique appliquées aux problèmes NP-difficiles. La boucle principale de la méthode consiste à réitérer l'exécution de recherches locales simples à partir de conditions initiales aléatoires. Le pseudo-code de la boucle principale est présenté dans l'Algorithme 6. Les opérations sont reparties en deux phases: une phase de construction suivie d'une phase d'amélioration. Dans l'Algorithme 6, une tentative de construction suivie d'une tentative d'amélioration sont effectuées par les deux appels *constructSolution* et *improveSolution* . Les détails de ces deux procédures sont donnés respectivement dans les sections 6.1.2 et 6.1.3. La construction génère rapidement des solutions partielles non nécessairement admissibles. Elle gère les dates d'émission des messages puis applique des opérateurs de construction de chemins. A partir de la solution obtenue, la procédure d'amélioration applique des modifications locales portant sur les chemins en vue d'améliorer la solution. L'amélioration suit des schémas de recherche locale spécifiés plus loin.

Dans tous les cas de figure, il convient d'évaluer et de comparer des solutions non nécessairement admissibles. C'est pourquoi, le classement des solutions s'effectue selon deux objectifs hiérarchisés qui sont le nombre de chemins admissibles (construits) et la longueur des chemins. Le premier objectif consiste à maximiser le nombre de chemins construits, tandis que le deuxième objectif consiste à minimiser la longueur des chemins. Le classement est effectué par la procédure *selectBest*. Il faut noter que la longueur des chemins correspond à la longueur totale dans la version min-sum du CKPP et du CRKPP, et correspond à la longueur du chemin le plus long dans la version min-max du même problème.

6.1.2 Boucle de construction

Ici, nous présentons dans l'Algorithme 6 la boucle principale inclue dans la procédure de construction *constructSolution*. La procédure *constructSolution* elle même est détaillée

Algorithme 6: Boucle principale de recherche locale itérée

Sorties : $Best$
début
  $S \leftarrow initialize()$; `// initialisation des structures de données et des dates d'émission`
  $count \leftarrow 0$;
  tant que $count < maxCount$ **faire**
    $count \leftarrow count + 1$;
    $S \leftarrow constructSolution(S)$; `// construction d'une solution non nécessairement admissible`
    $S \leftarrow improveSolution(S)$; `// amélioration de la solution par recherche locale`
    $Best \leftarrow selectBest(S, Best)$;
  retourner $Best$;

dans l'Algorithme 7. Son rôle est de générer aussi rapidement que possible de nouvelles solutions candidates, admissibles ou non, présentant un nombre maximum de chemins construits. Pour ce faire, la procédure répète *maxConstruct* fois un procédé de construction de base. La gestion des dates d'émission des messages se situe dans les premières opérations à effectuer. Une fois fixées les dates d'émission, l'opérateur de construction parallèle gloutonne tente de générer simultanément K chemins sans conflits le plus rapidement possible. Ensuite, l'algorithme de Dijkstra modifié, qui prend plus de temps, tente une reconstruction des chemins « bloqués » résiduels un à un. La procédure *selectBest* effectue un classement suivant les deux objectifs hiérarchisés que sont le nombre de chemins construits et la longueur (totale ou maximum) des chemins.

6.1.3 Boucle d'amélioration

La génération de nouvelles solutions de départ diversifiées à partir de conditions initiales aléatoires est effectuée par la boucle de construction. À partir de la solution fournie, l'intensification de la recherche est réalisée par la procédure *improveSolution* inclue dans l'Algorithme 6. Elle permet des mouvements de plus faible ampleur dans une petite région de l'espace des solutions, en quelques pas, et essaie de transformer des solutions non admissibles en solutions admissibles et de les améliorer. Le schéma de base d'une procédure d'amélioration consiste à intégrer un opérateur de voisinage dans une stratégie de recherche. En suivant ce schéma, nous avons dégagé trois versions de recherche locale mettant en œuvre la procédure d'amélioration. L'opérateur de voisinage utilisé, qui permet une modification locale élémentaire de la solution courante, est dans tous les cas de figure celui de la section 5.4 du chapitre 5. Il consiste à supprimer puis reconstruire un sous-ensemble de chemins de la solution. Le nombre de chemins supprimés/reconstruits détermine la taille du voisinage. Ce paramètre ne doit

Algorithme 7: constructSolution

Entrées : $S \in$ Solution, $maxConstruct$
Sorties : $Best$
début
  $count \leftarrow 0$;
  tant que $count < maxConstruct$ **faire**
    $count \leftarrow count + 1$;
    `// Appliquer la permutation des dates avec une probabilité de 0.5`
    si $rand(0,1) > 0.5$ **alors**
      $S \leftarrow permuteDepartureDates(S)$;
    $S \leftarrow translateDepartureDates(S)$;
    Initialiser le TEG; `// Tous les arcs sont mis à « libre » pour tous les slots de temps`
    Supprimer tous les chemins de S;
    $S \leftarrow constructSolutionPar(S)$; `// Construction parallèle gloutonne`
    $S \leftarrow constructSolutionWithModifiedDijkstra(S)$; `// Reconstruction des chemins bloqués un-à-un par Dijkstra dans le TEG`
    $Best \leftarrow selectBest(S, Best)$;
  retourner $Best$;

pas être confondu avec la taille de l'échantillon, qui correspond au nombre de solutions examinées dans le voisinage de façon stochastique dans certaines recherches locales. Trois versions de recherche locales sont proposées : une recherche aléatoire simple et deux recherches locales à pivotage.

6.1.3.1 Recherche aléatoire itérée

La stratégie de recherche définie par l'Algorithme 8 s'inspire du principe de la marche aléatoire. Nous l'avons dénommée recherche aléatoire itérée ou IRS. Elle consiste à faire évoluer une solution courante en effectuant un nombre donné de mouvements successifs dans le voisinage puis à sélectionner la meilleure solution rencontrée lors de cette succession de mouvements. La méthode est très simple car très peu d'opérations de copie de données sont effectuées à chaque pas d'itération.

6.1.3.2 Recherches locales *first improvement* et *best improvement*

Dans cette section, nous présentons l'algorithme des deux stratégies de la méthode de recherche locale. La première est la recherche locale gloutonne, que nous appelons « recherche locale itérée premier meilleur » , ou encore ILS-FI. La seconde est la recherche locale en profondeur, que nous appelons « recherche locale itérée meilleure amélioration » , ou encore ILS-BI. L'Algorithme 9 donne le code commun aux deux versions. La différence entre les deux exécutions tient dans la condition d'évaluation de la boucle interne « tant que » . Le code correspond à la version ILS-BI. L'instruction en commentaire dans la condition, qui test la détection d'une amélioration, doit être ajoutée dans le cas ILS-FI. Par rapport à IRS, ILS-FI et ILS-BI comportent chacune deux boucles imbriquées, et non une seule, comme le montre l'Algorithme 9. La boucle externe contrôle la profondeur de la recherche. L'algorithme s'arrête lorsqu'aucune amélioration n'a été trouvée, ce qui correspond à l'atteinte d'un minimum local. La solution courante constitue l'élément pivot autour duquel s'effectue la recherche dans le voisinage. C'est à partir de cet élément que des solutions voisines sont générées et examinées. La boucle interne met en œuvre la règle de pivotage. Elle détermine la meilleure solution voisine vers laquelle la recherche doit se déplacer. Cette solution devient le nouveau pivot à partir duquel des solutions voisines sont à nouveau générées.

Par rapport à la recherche aléatoire itérée IRS, nous pouvons noter l'ajout d'une variable supplémentaire S' utilisée pour tester la présence d'une amélioration. Dans les deux types de recherche locale ILS-FI et ILS-BI, une variable supplémentaire de mémorisation est donc requise pour tester la possibilité d'amélioration, cela entraîne davantage de copies de données que dans la première méthode de recherche aléatoire itérée IRS. Dans ILS-FI, la première meilleure solution rencontrée devient le nouvel élément pivot. Dans ILS-BI, c'est la meilleure solution dans un échantillon aléatoire du voisinage qui devient le nouvel élément pivot. La taille de l'échantillon est définie par le paramètre *neighborhoodSampleSize* dans l'Algorithme 9. Dans les expérimentations, nous avons fixé à 100 la taille de l'échantillon.

Algorithme 8: iteratedRandomSearch

Entrées : $S \in$ Solution, $maxImprove$, K Nombre de messages
Sorties : $Best$
début
  $count \leftarrow 0$;
  tant que $count < maxImprove$ **faire**
    $count \leftarrow count + 1$;
    $S \leftarrow generateNeighbor(S, rand(1, K))$;
    $Best \leftarrow selectBest(S, Best)$;
  retourner $Best$;

Algorithme 9: Recherche locale premier-meilleur-voisin, meilleur-voisin

Entrées : $S \in$ Solution, K Nombre de messages
Sorties : $Best$
début
  $Best \leftarrow S$;
  $improvementFound \leftarrow true$;
  tant que $improvementFound$ `/* définit la profondeur */` **faire**
    $count \leftarrow 0$;
    $improvementFound \leftarrow false$;
    tant que
    $count < neighborhoodSampleSize$ `/*` **et non** `improvementFound */`
    `/* règle de pivotage BI ou FI */` **faire**
      $count \leftarrow count + 1$;
      $S' \leftarrow generateNeighbor(S, rand(1, K))$ `/* examen du voisin */`;
      si $isBest(S', Best)$ **alors**
        $Best \leftarrow S'$;
        $improvementFound \leftarrow true$;
    $S \leftarrow Best$;
  retourner $Best$;

6.2 Algorithme évolutionnaire et algorithme mémétique

6.2.1 Boucle principale de l'algorithme évolutionnaire/mémétique

Suivant la terminologie des algorithmes évolutionnaires, les solutions sont maintenant des « individus » composant une population qui va évoluer de génération en génération de manière à répondre aux exigences du problème. Des opérateurs de variations tels que des mutations vont modifier les individus, ceux-ci étant soumis à une sélection suivant leur adéquation aux objectifs du problème. Ici, nous spécifions une boucle évolutionnaire principale commune aux deux versions d'algorithme évolutionnaire que nous considérons dans ce travail. Nous déclinerons cette boucle principale selon un algorithme évolutionnaire classique et un algorithme de type algorithme mémétique. Un algorithme mémétique est une extension d'un algorithme génétique ou évolutionnaire dans lequel est ajouté une recherche locale en tant qu'opérateur de modification de solution correspondant à une mutation d'un type particulier [49]. La différence entre les deux approches évolutionnaire et mémétique proposées dans notre travail réside dans l'instanciation des différentes opérations de mutation. Celles-ci sont atomiques et brèves dans le premier cas, tandis qu'elles sont réalisées par des processus de recherche plus complexes dans le deuxième cas. Nous n'utilisons pas d'opérateur de croisement. Dans l'algorithme mémétique, nous remplaçons les opérations élémentaires de

voisinage de la version dite évolutionnaire, par des recherches locales. Ainsi, nous réutilisons nos méthodes de recherche locale indépendantes en tant qu'opérateurs de mutation dans la boucle évolutionnaire.

Une solution consiste avant tout de déterminer une date d'émission dans l'intervalle $[0, T-1]$ pour chaque message, et l'ensemble des chemins origine/destination sans conflit pour l'ensemble des messages. La solution est admissible lorsque l'ensemble des chemins de tous les messages est construit. Et non admissible lorsqu'au moins un chemin d'un message n'a pas été construit. Pour évaluer la qualité d'une solution, ou d'un individu, une fonction appelée *fitness* associe une valeur scalaire à chaque individu de la population. Les « meilleurs » individus ont la plus grande fitness, tandis que les « mauvais » individus une fitness plus faible. Pour une solution S donnée, elle est définie par $fitness(S) = nbuilt - 10^{-\lceil log_{10}(Max) \rceil} \times length$, où $nbuilt$ est le nombre de chemins construits, Max une limite supérieure de la longueur totale maximum des chemins, et $length$ la longueur totale (somme des longueurs) des chemins. De cette façon, le nombre de chemins construits est considéré comme le premier objectif (à maximiser), et la longueur totale comme un objectif secondaire (à minimiser). On notera que la prise en compte de cette valeur de fitness est équivalente à une comparaison lexicographique portant sur les deux objectifs. Pour des raisons de précisions sur les comparaisons en nombres flottants, nous préférerons généralement employer une comparaison lexicographique directe portant sur les objectifs définis en nombres entiers.

Le pseudo-code de la boucle évolutionnaire commune est donné par l'Algorithme 11. Deux opérateurs de sélection par rang opérant au niveau de la population sont utilisés. Le premier d'entre eux est l'opérateur appelé *select* dans le pseudo-code. Il remplace $Pop/5$ individus de plus faible *fitness* dans la population par $Pop/5$ individus de plus grande *fitness*. Avec Pop comme taille de la population d'individus. Le second opérateur de sélection et de classement est la version élitiste du premier. Il est appelé *selectElitist*. Il remplace $Pop/10$ individus ayant la *fitness* la plus faible de la population, par le meilleur individu $Best1$ rencontré durant l'exécution. Deux opérateurs de mutations sont spécifiés. Ceux sont les opérateurs *mutate* et *localSearch*. Le premier est chargé de la manipulation et gestion des dates d'émission. Le deuxième correspond à des mouvements de voisinage, les dates d'émission étant fixées. Pour éviter une stagnation dans un minimum local, un opérateur de régénération *generate* est appliqué avec une faible probabilité. Il effectue une régénération de toute la population et réinitialise le meilleur individu rencontré $Best1$. Notons que toute modification sur les dates d'émission entraîne une refonte complète des chemins origine/destination, tandis qu'un mouvement de voisinage porte sur une suppression/reconstruction d'un sous-ensemble de chemins sans modification des dates. Nous précisons l'instanciation de ces opérateurs pour les algorithmes évolutionnaire et mémétique dans les deux sections suivantes. Il faut noter qu'en considérant la longueur totale comme objectif (secondaire) à minimiser dans la fonction de *fitness*, nous traitons la version min-sum du CKPP et du CRKPP, qui est la version par défaut. La version min-max est obtenue

en considérant la longueur du chemin le plus long comme objectif (secondaire) à minimiser en lieu et place de la longueur totale.

Algorithme 10: Boucle principale de l'algorithme évolutionnaire et mémétique

Sorties : $Best2$
début
  $Best1, Best2 \in$ Solution;
  P: Population;// 100 individus
  $Gen \in$ **Entier**;
  $Gen \leftarrow 0$;
  $P \leftarrow generate(P)$; // Générer les individus
  $Best1 \leftarrow getBest(P)$;
  // Répéter $MaxGen$ générations
  tant que $Gen < MaxGen$ **et** $\neg$ *(solution construite)* **faire**
    $Gen \leftarrow Gen + 1$;
    // Appliquer une opération/mutation de voisinage
    $P \leftarrow locaSearch(P)$;
    // Mémoriser le meilleur individu
    $Best1 \leftarrow getBest(P, Best1)$;
    $Best2 \leftarrow getBest(Best1, Best2)$;
    // Appliquer les opérateurs de sélection
    $P \leftarrow select(P, size(P)/5)$;
    $P \leftarrow selectElitist(P, Best1, size(P)/10)$;
    // Appliquer une mutation
    $P \leftarrow mutate(P)$;
    // Appliquer une régénération avec une probabilité très faible
    si $rand(0, 1) > 0.99$ **alors**
      $P \leftarrow generate(P)$; // Régénérer les individus et le meilleur individu
      $Best1 \leftarrow getBest(P)$;
  retourner $Best2$;

6.2.2 Algorithme évolutionnaire

Le détail des opérateurs de l'approche de type évolutionnaire classique est donné dans l'Algorithme 11. Les opérations sont élémentaires parce qu'elles consistent en des appels directs aux opérateurs de base. La multiplication des opérations locales et de diversification sur les individus couplée à la dynamique de sélection permet ainsi une recherche stochastique distribuée sur l'espace des solutions. Notons que les dates d'émissions évoluent au moyen des opérateurs de permutation et de translation (opéra-

teur de gestion des dates d'émission) et que la permutation est appliquée avec une probabilité de 0.5. L'opérateur *localSearch* effectue ici un simple mouvement élémentaire dans le voisinage par un appel direct à l'opérateur de voisinage.

Algorithme 11: Les opérateurs de variation de l'algorithme évolutionnaire

Procédure **generate**(P: Population)
début
  pour *chaque individu* $I \in P$ **faire**
    $I \leftarrow permuteDepartureDates(I)$;
    $I \leftarrow translateDepartureDates(I)$;
    Initialiser le TEG; `// Tous les arcs sont mis à « libre » pour tous les slots de temps`
    Supprimer tous les chemins de I;
    $I \leftarrow constructSolutionPar(I)$; `// Construction parallèle gloutonne`
    $I \leftarrow constructSolutionWithModifiedDijkstra(I)$; `// Reconstruction des chemins bloqués un-à-un par Dijkstra dans le TEG`

Procédure **mutation**(P: Population)
début
  pour *chaque individu* $I \in P$ **faire**
    `// Appliquer la mutation avec une probabilité de 0.5`
    si $rand(0,1) > 0.5$ **alors**
      Initialiser le TEG; `// Tous les arcs sont mis à « libre » pour tous les slots de temps`
      `// Gérer les dates d'émission`
      $I \leftarrow permuteMessages(I)$;
      $I \leftarrow translateDepartureDates(I)$;

Procédure **localSearch**(P: Population)
début
  pour *chaque individu* $I \in P$ **faire**
    $I \leftarrow generateNeighbor(I, rand(1,K))$;

6.2.3 Algorithme mémétique

Le détail des opérateurs de mutation pour l'algorithme de type mémétique est donné dans l'Algorithme 12. Dans ce cas, les opérations de mutation deviennent des appels aux procédures de construction et de recherche locale que nous avons spécifiées antérieurement dans ce chapitre dans la section sur les recherches locales itérées. La procédure d'amélioration que nous utilisons dans les expérimentations est la recherche aléatoire itérée, appelée précédemment méthode IRS, du fait qu'elle fixe un nombre constant d'itérations. Les opérations élémentaires brèves et distribuées sur l'espace des

solutions réalisées dans l'algorithme évolutionnaire précédent deviennent des processus longs et plus coûteux en temps de calcul. L'opérateur *mutation* effectue un appel de la procédure de construction itérée tandis que l'opérateur *localSearch* exécute un appel de la procédure de recherche locale. L'intérêt recherché est de coupler la dynamique de diversification et de sélection de l'approche évolutionnaire avec des recherches locales adaptées pour l'intensification de la recherche dans une zone réduite de l'espace des solutions.

Algorithme 12: Les opérateurs de variation de l'algorithme mémétique

```
Procédure generate(P ∈ Population)
début
    pour chaque individu I ∈ P faire
        I ← constructSolution(I) // Correspond à la construction
            spécifiée dans l'Algorithme (7)
Procédure mutation(P ∈ Population)
début
    pour chaque individu I ∈ P faire
        // Appliquer la mutation avec une probabilité de 0.5
        si rand(0, 1) > 0.5 alors
            I ← constructSolution(I) // Correspond à la construction
                spécifiée dans l'Algorithme (7)
Procédurelocalsearch(P ∈ Population)
début
    pour chaque individu I ∈ P faire
        I ← improveSolution(I) // Correspond à une des recherches
            locales spécifiées aux Algorithmes (8-9)
```

6.3 Conclusion

Nous avons présenté l'ensemble de nos méthodes de résolution pour les problèmes standard et reconfigurable, du routage dans le NoC, dont nous avons proposé une formalisation sous la forme d'un problème d'optimisation combinatoire. De façon générale, ces méthodes se justifient par la NP-difficultés des problèmes de routage, et de l'incapacité de la méthode exacte utilisée à prendre en charge les grandes instances du problème. L'évolution de ces méthodes avec des approches opérants sur une solution unique vers des approches opérants sur une population de solutions, leur confèrent la capacité de résolution des problèmes de routage de façon générale. Les trois versions de la recherche locale n'opèrent que sur le problème de routage GT standard. Leur différence tient au mode d'examen du voisinage retenu. La première s'inspirant de la

recherche aléatoire de type marche aléatoire, la deuxième, une recherche gloutonne et la troisième, une recherche en profondeur. L'algorithme évolutionnaire permet de repousser les limites des méthodes de recherche locale, par une extension, en intégrant une approche à base de population de solutions et d'opérateurs de sélection, au cas reconfigurable du problème de routage. L'algorithme mémétique, permet un couplage de la dynamique de diversification et de sélection de l'approche évolutionnaire, avec des recherches locales adaptées pour l'intensification dans une zone réduite de l'espace de solutions. Il tire ainsi partie des avantages de la recherche locale et de l'algorithme évolutionnaire. Tandis que les approches de recherches locales opèrent uniquement sur le problème de routage GT standard, les algorithmes évolutionnaire et mémétique peuvent opérer aussi bien sur le problème de routage GT standard que sur celui de son extension au cas reconfigurable. Nous allons maintenant procéder aux évaluations des performances de ces approches sur des jeux de tests de différentes tailles. Ceux-ci comportent des instances structurées issues de cas réels d'application et des instances non structurées générées aléatoirement.

7 Présentation des jeux de tests

L'évaluation des heuristiques nécessite l'utilisation d'instances de problème représentatives de la difficulté et de la diversité des cas réels d'application. Pour évaluer nos méthodes de résolution, nous utilisons six instances de jeux de tests structurés des problèmes. Ceux-ci sont extraits d'applications réelles mises en œuvre et étudiées dans [9]. Ces jeux de tests structurés ont des topologies de taille croissante et peuvent comporter 12 messages pour la plus petite à 209 messages pour la plus grande. Afin d'approfondir l'étude de la robustesse des algorithmes dans des cas divers et variés de condition de trafic, nous proposons en supplément de ces instances de jeux de tests structurés des problèmes, un générateur d'instances de jeux de trafic aléatoire. Selon une spécification de topologie de NoC, et des canaux de communication ouverts entre IP, il permet de générer très rapidement de façon aléatoire uniforme des instances des jeux de trafic répondant aux spécifications des flots de données telles que présentées dans les chapitres 2 et 3. Il fonctionne aussi bien pour le cas du routage standard sans reconfiguration que pour le cas avec reconfiguration dynamique. Une topologie particulière et ses spécifications de tables TDMA est dédiée au problème de routage avec reconfiguration dynamique.

Ce chapitre est exclusivement dédié à la présentation des jeux de tests structurés et à la description du générateur d'instances aléatoires. Les évaluations de performances proprement dites seront présentées dans les deux chapitres 7 et 8. Le chapitre8 présentera l'évaluation des recherches locales sur le problème standard. Le 9 chapitre traitera de l'approche évolutionnaire et de son application au problème avec reconfiguration dynamique. Dans ce chapitre, les jeux de tests structurés sont décrits dans la section 7.1 et le générateur d'instances aléatoires dans la section 7.2 selon les cas standard et reconfigurable. La section 7.3 donne quelques généralités sur la conduite des expérimentations et une conclusion termine le chapitre.

7.1 Jeux de tests structurés issus de cas réels d'application

A partir de quatre topologies de NoC, six instances de problèmes issus de cas réels d'application ont été proposés par [9]. Ces jeux de tests structurés sont utilisés ici comme cas types d'application représentatifs de la difficulté du problème. Ils servi-

ront de *template* pour la génération de jeux de trafic aléatoires. Les quatre types de topologies sont nommés N1, N2, N3, et N4. Ils correspondent respectivement à des NoC de tailles croissantes. À partir de ces quatre topologies, six instances de problème sont fournies. Elles sont nommées N1, N2, N3A, N3B, N3AB et N4 en reprenant le nom de la topologie correspondante. La topologie N3 est utilisée sous trois formes de problème. Les deux instances N3A et N3B correspondent respectivement à deux ensembles de tables TDMA A et B avec leur spécification de messages, tandis que l'instance N3AB correspond à une instance de communication reconfigurable permettant l'échange dynamique des tables des groupes A et B. Seule l'instance N3AB sert de base à l'évaluation du problème avec reconfiguration dynamique. Cette instance est de taille moyenne, mais est suffisamment large selon nous pour mettre en évidence la difficulté de résolution du problème avec reconfiguration. Des jeux de trafic aléatoires en sont aussi dérivés.

Chaque instance est caractérisée par quatre paramètres qui sont la longueur en *time-slots* du cycle d'émission T, le nombre N de routeurs, le nombre P de composants IP, c'est-à-dire de nœuds source/destination, et le nombre K de messages.

Le jeu de test N1 est la plus petite instance. Il est détaillé par la figure 7.1. En plus des quatre paramètres qui définissent la taille de l'instance, sont précisées les spécifications des tables TDMA définissant les messages origine/destination avec les nombres de paquets correspondants. Par exemple, l'IP 0 émet trois messages de deux paquets chacun, respectivement vers les trois IP d'identifiants 1, 3, et 2. Ainsi, la taille des tables TDMA de cette instance, qui définit le cycle de temps d'émission, est de $T = 6$ slots de temps. Le nombre de routeurs est de $N = 7$ et ils connectent $P = 4$ composants émetteur/récepteur. De plus, la figure donne des exemples de chemins possibles origine/destination pour les $K = 12$ messages au total.

Le deuxième jeu de test structuré, nommé test N2, est spécifié par la figure 7.2. Le jeu de test comporte $N = 8$ routeurs pour $P = 7$ composants émetteur/récepteur. Les tables TDMA sont de longueur $T = 8$ intervalles de temps et spécifient au total $K = 28$ messages origine/destination. Par exemple, le composant noté IP1 sur la figure émet quatre messages de deux slots de temps chacun, respectivement vers les composants notés M1, M2, M3, et IP2 sur la figure. Cette instance est de taille supérieure à la précédente.

Le jeu de test structuré de la figure 7.3 nommé N3 est décliné en trois instances différentes, dont l'une spécifie un jeu de trafic avec reconfiguration dynamique. Il connecte $P = 10$ composants émetteurs/récepteur bar le biais d'un réseau avec $N = 15$ routeurs. La taille du cycle d'émission, et donc des tables TDMA, est de $T = 9$ slots de temps. Suivant la configuration des tables TDMA de ses composants émetteurs, nous pouvons en déduire trois instances de problème. Les composants émetteurs IP0, IP1 et IP2 possèdent chacun deux tables TDMA, spécifiées l'une au dessus de l'autre et proche du composant sur la figure. Les tables spécifiées au dessus constituent un premier groupe noté A, tandis que les tables spécifiées au dessous constituent un deuxième groupe de table TDMA noté B. A ces deux groupes correspondent respective-

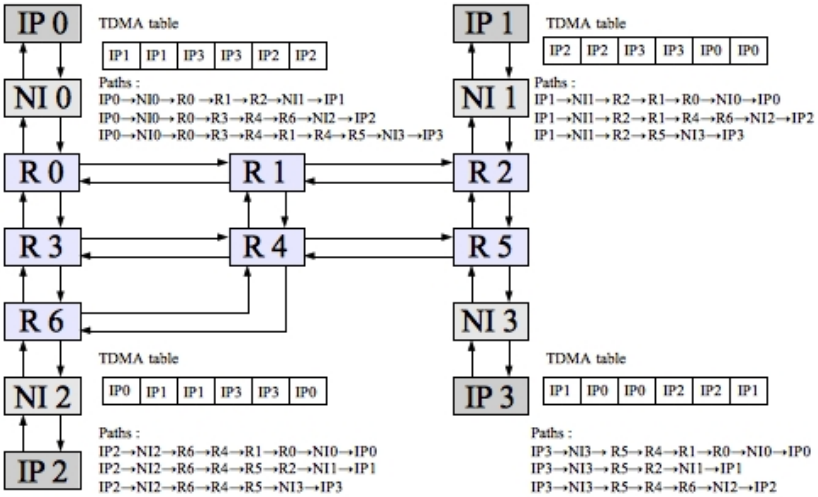


Figure 7.1: Instance N1, avec $T = 6, N = 7, P = 4, K = 12$.

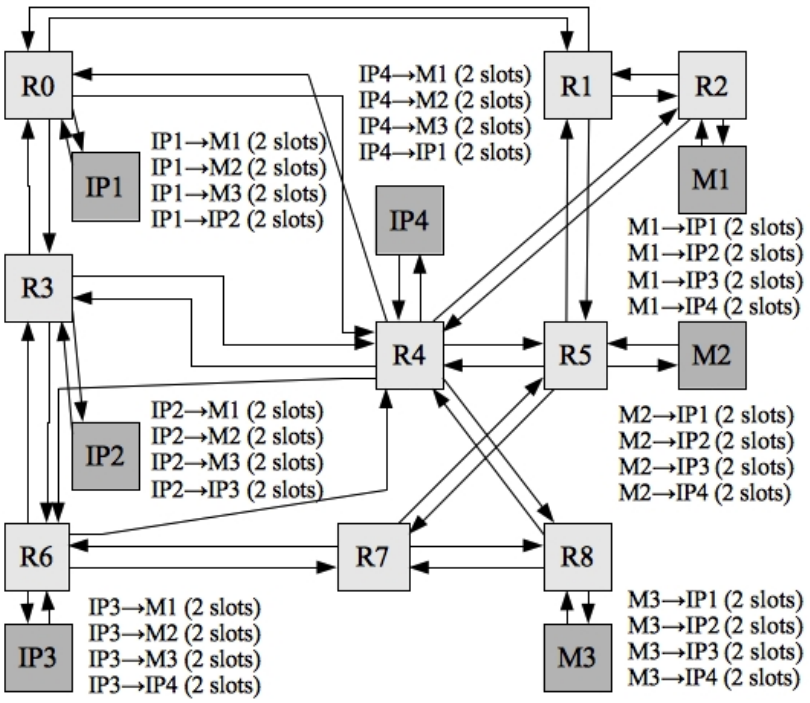


Figure 7.2: Instance N2, avec $T = 8, N = 9, P = 7, K = 28$.

ment deux instances notées N3A et N3B du problème standard (CKPP). Une instance du problème avec reconfiguration dynamique (CRKPP) notée N3AB est obtenue en

considérant l'utilisation conjointe des tables des groupes A et B. Dans ce cas, chaque émetteur parmi IP0, IP1 et IP2 peut échanger l'application d'une de ses deux tables TDMA qui lui sont affectées à chaque cycle de temps, selon ses besoins en communication. Les nombres de messages des trois instances N3A, N3B, et N3AB sont respectivement de $K = 27$, $K = 24$ et $K = 33$ messages. On peut noter que résoudre l'instance N3AB du problème dynamique revient indirectement à résoudre 2^3 instances distinctes du problème standard, puisque chaque table TDMA peut être interchangée par un émetteur de manière asynchrone indépendamment du choix effectué par les autres émetteurs. Le but de notre travail est de permettre une réutilisation maximale des slots de temps laissés vacants lors des permutations de tables ainsi qu'autorisé par le théorème (3) du chapitre 4.

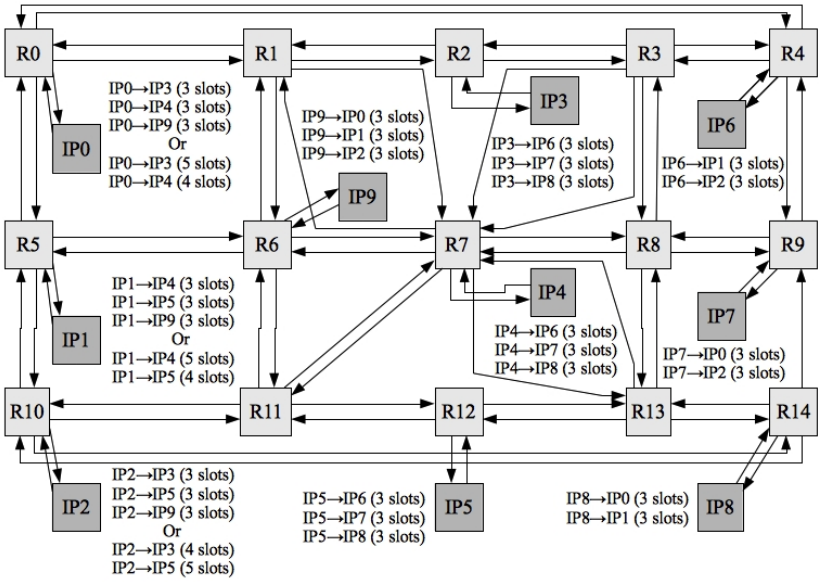


Figure 7.3: Instances N3A, N3B, et N3AB, avec $T = 9, N = 15, P = 10, K = 27/24/33$.

Le plus grand de nos jeux de tests structurés utilisé pour le problème standard est spécifié dans les trois figures 7.4-7.6. La figure 7.4 fournit la topologie du réseau, tandis que les deux figures 7.5 et 7.6 détaillent les messages origine/destination avec leurs paquets correspondants. Le cycle d'émission est de $T = 47$ intervalles de temps. Le NoC comporte $N = 36$ routeurs pour $P = 35$ composants émetteur/récepteur. Le nombre de messages est $K = 209$. Ce jeu de test est de taille relativement importante. Il permettra d'évaluer le comportement des heuristiques en fonction de la taille du problème. Il servira également à illustrer leur comportement en fonction de la saturation en trafic des émetteur/récepteur. Nous verrons que le générateur aléatoire de trafic permet d'obtenir, dans ce cas particulier de topologie, des configurations de flot de données davantage saturées que celle fournie par R. Dafali en tant qu'application réelle type. Le

taux de saturation est défini par le critère appelé TSL au chapitre 2.

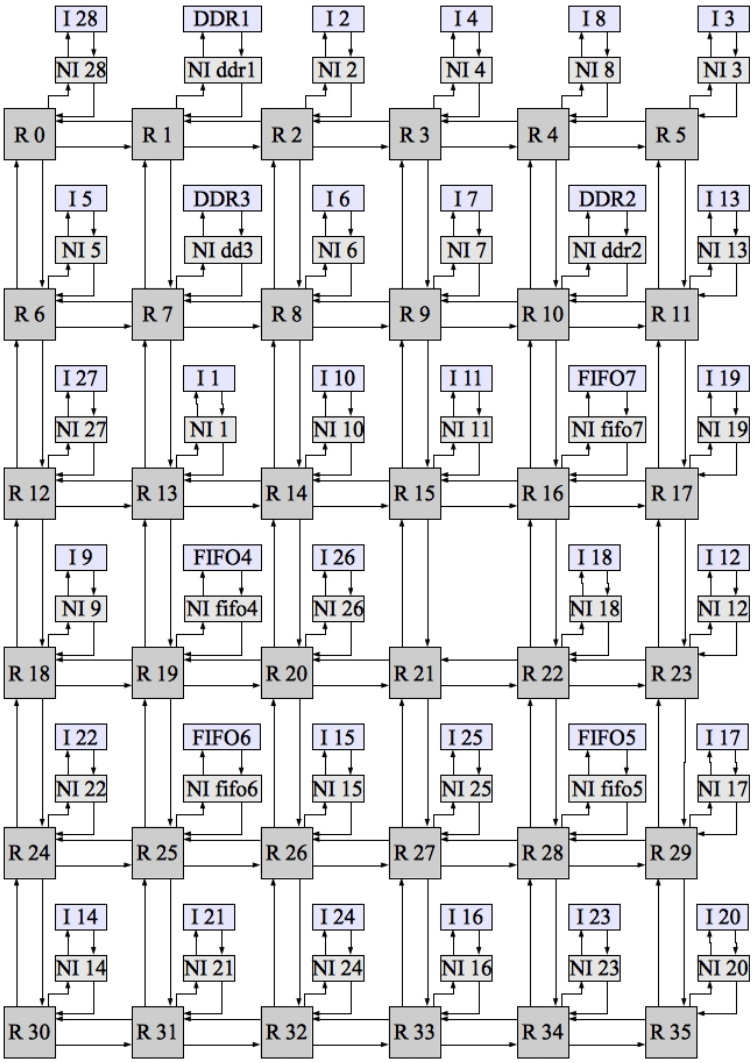


Figure 7.4: Instance N4, avec $T = 47, N = 36, P = 35, K = 209$.

7.2 Générateur de jeux de trafic aléatoires

Pour évaluer la robustesse de nos méthodes de recherche sur des configurations variées de flots de communication, nous avons développé un générateur d'instances aléatoires. À partir du graphe de communication de l'application, qui spécifie les canaux de communication ouverts entre IP, le générateur alloue aux communications un nombre de

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13	I14	I15	I16	I17	I18	I19	I20	I21	I22	I23	I24	I25
I1																									
I2																									
I3																									
I4																									
I5																									
I6																									
I7																									
I8																									
I9																									
I10																									
I11																									
I12																									
I13																									
I14																									
I15																									
I16																									
I17																									
I18																									
I19																									
I20																									
I21																									
I22																									
I23																									
I24																									
I25																									
I26																									
I27																									
I28																									
DDR1	2	2	2	2	2	2		2	2	2	2	2	2	2	2	2	2	2	2						
DDR2	2	2	3	2	2		2	10	2	2	2	2	2	2	2	2	2	2	2						
DDR3	2	2	2	2	2	2	3	2	2	2	2	2	2	2	2	2	2	2	2						
FIFO4	2								2	2		2		2	2	2	2	2	2			2	2	2	2
FIFO5																2	2		2	2			2		2
FIFO6														2	2			2			2	2		2	
FIFO7	2									2	2	2		2	2	2	2	2	2	2	2	2	2	2	2
	10	6	7	6	6	4	5	14	8	10	8	10	6	12	12	12	12	12	12	4	4	6	6	6	6

Figure 7.5: Spécification des messages et leurs tailles pour le cas N4.

paquets qui définit la bande passante de chaque communication origine/destination. Des jeux de trafic aléatoires sont produits aussi bien pour le problème standard CKPP que pour le problème avec reconfiguration dynamique CRKPP. Il faut noter que les flots de communication produits sont conformes aux spécifications énoncées dans les sections 2-2.4 et 3-3.3 respectivement pour les cas standard et reconfigurable du problème. Le but est de générer rapidement des solutions variées sous-optimales des problèmes de flots spécifiés, aussi saturées que possible en nombre de paquets émis. Pour rappel, la figure 7.7 donne un exemple de graphe de communication en (a), constituant la donnée d'entrée du générateur, et un exemple de jeu de trafic en (b), constituant la donnée de sortie. Le cas de trafic avec reconfiguration dynamique est une extension du problème standard, dans laquelle la matrice de communication origine/destination est une matrice multi-ligne. Les colonnes décrivent les récepteurs, tandis que les lignes décrivent les émetteurs, un même émetteur pouvant être configuré par plusieurs lignes de la matrice correspondant à ses différentes tables TDMA.

Pour générer une instance de trafic aléatoirement et rapidement, l'algorithme procède en deux étapes qui portent respectivement sur l'allocation de paquets en réception

I26	I27	I28	DDR1	DDR2	DDR3	FIFO4	FIFO5	FIFO6	FIFO7	
			2	2	2	2			2	10
			2	2	2					6
			2	2	2					6
			2	2	2					6
			2	2	2					6
			2		2					4
			2	2	2					6
			2	2	2					6
			2	2	2	2				8
			2	2	2	2			2	10
			2	2	2				2	8
			2	2	2	2			2	10
			2	2	2					6
			2	2	2	2		2	2	12
			2	2	2	2		2	2	12
			2	2	2	2	2		2	12
			2	2	2	2	2		2	12
			2	2	2	2		2	2	12
			2	2	2	2	2		2	12
							2		2	4
								2	2	4
						2		2	2	6
						2	2		2	6
						2		2	2	6
						2	2		2	6
						2			2	4
						2			2	4
			2	2	2					6
		2								38
		2								47
		2								41
2	2									32
										12
										12
2	2									36
4	4	6	40	38	40	32	12	12	36	

Figure 7.6: Spécification des messages et leurs tailles pour le cas N4 (suite et fin).

et en émission. Lors de la première étape, on procède à une répartition aléatoire de la bande passante de chaque récepteur, un à un, entre les différents émetteurs qui lui sont associés. Les valeurs obtenues pour chaque nœud récepteur déterminent le trafic maximum qu'un nœud émetteur donné peut envoyer à ce récepteur, quelle que soit la configuration de TDMA considérée par l'émetteur. Ainsi, dans cette première étape,

	tdma	IP3	IP4	IP5
IP0	tdma 0	0	1	1
	tdma 1	1	0	1
IP1	tdma 0	1	0	1
IP2	tdma 0	1	1	0
	tdma 1	1	0	0

(a)

	tdma	IP3	IP4	IP5	total
IP0	tdma 0	0	T/2	T/2	T
	tdma 1	T/8	0	3T/4	7T/8
	max	T/8	T/2	3T/4	T
IP1	tdma 0	T/8	0	T/4	3T/8
	max	T/8	0	T/4	3T/8
IP2	tdma 0	T/2	T/2	0	T
	tdma 1	3T/4	0	0	3T/4
	max	3T/4	T/2	0	T
total max		T	T	T	3T 19T/8

(b)

Figure 7.7: Un ACG (a) et une configuration de flot reconfigurable (b).

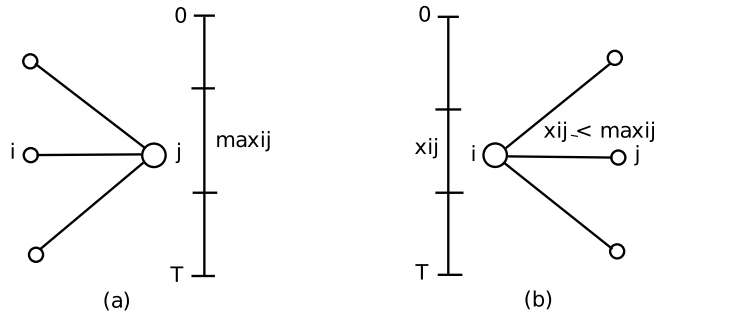


Figure 7.8: (a) Partitionnement de la bande passante en réception.(b) Ajustement de la bande passante en émission.

illustrée par le cas (a) de la figure 7.8, il n'est autorisé la réception par le récepteur j que d'au plus maxij paquets en provenance de l'émetteur i. Et cela, quelque soit la configuration de TDMA de l'émetteur i. Il convient de s'assurer que les valeurs maximales attribuées selon les colonnes (récepteurs) vérifient les contraintes de capacité énoncées dans la spécification. Dans une deuxième étape, on ajuste les bandes passantes de chaque émetteur, cela pour chaque table TDMA, tout en essayant de maximiser le nombre de paquets envoyés en respectant les contraintes de capacité énoncées dans la spécification des flots de données. La deuxième étape est illustrée par le cas (b) de la figure 7.8, qui amène le nombre de paquets xij, envoyés par n'importe quel TDMA de l'émetteur i vers le récepteur j, à être au plus égale à maxij. L'Algorithme 13 spécifie succinctement les étapes de la génération d'instances aléatoires. A chaque étape, une

Algorithme 13: Génération de jeux de trafic aléatoires

Entrées : Graphe de communication de l'application
Sorties : Instance aléatoire
début
  pour *chaque nœud récepteur* $j \in P$*: ensemble des composants émetteur/récepteur* **faire**
    $columnMaxCalcul(j, T, seuilTraficParam)$; `// Allouer des valeurs maximales de paquets reçus pour chaque récepteur` j`, le seuil minimum d'allocation étant pris en compte.`
    $improveColumnMaxCalcul(j)$; `// Ajustement des valeurs maximums pour tenir compte des arrondis et maximiser les valeurs maximum associées au récepteur` j`.`
  pour *chaque nœud émetteur* $i \in P$*: ensemble des composants émetteur/récepteur* **faire**
    $lineTimeSlotSearch(i, T, seuilTraficParam)$; `// Attribuer des paquets à l'émetteur` i `et ses différentes TDMA en tenant compte des seuils maximaux affectés à chaque récepteur, et en tenant compte du seuil minimum.`
    $improveLineTimeSlotSearch(i)$; `// Ajustement des valeurs de paquets de l'émetteur` i `pour tenir compte des arrondis et maximiser les valeurs attribuées.`
  retourner S;

difficulté se pose parce que les valeurs de trafic sont des valeurs entières plutôt que des valeurs fractionnaires. Par conséquent, les valeurs décimales sont arrondies à la valeur entière inférieure la plus proche, et les paquets perdus sont réinjectées au hasard, un à un, dans les différentes communications, en prenant soin de ne pas dépasser le nombre maximal de slots de temps autorisés entre chaque paire de nœuds émetteur/récepteur. Les choix aléatoires tiennent aussi compte du seuil minimum s de trafic, de telle façon qu'au moins s paquets sont envoyés par chaque nœud émetteur. En règle générale, nous avons fixé ce seuil à $s = 2$ paquets pour les expérimentations présentées, puisque les messages contiennent au minimum un paquet d'entête suivi d'un paquet de donnée. Afin d'évaluer le niveau de saturation en trafic du NoC, nous rappelons que nous utilisons le *Traffic Saturation Level* (TSL) défini par l'équation (2.16) du chapitre 2 pour le cas du problème standard, et le *Maximum Reception Throughput* (MRT) défini par l'équation (3.7) du chapitre 3 pour le cas du problème avec reconfiguration dynamique. Par exemple, l'instance N3AB prévue pour tester la reconfiguration dynamique présente un TSL de 90% pour un MRT de 100%. Le MRT est une extension du TSL permettant de prendre en compte des niveaux de trafic à différents moments selon les configurations possibles de tables TDMA pour un émetteur.

7.3 Démarche d'évaluation pratique

Les heuristiques proposées sont développées en C++ et ont été exécutées sur un PC Intel Core Duo 3.0 GHz, avec un seul cœur utilisé. Toutes les évaluations sont effectuées en moyenne sur une base de 100 exécutions par instance de problème. Pour chaque test, nous évaluons la longueur totale des chemins, et la longueur moyenne d'un chemin, en divisant la longueur totale par le nombre de messages K. De même, nous évaluons le temps de calcul moyen sur 100 exécutions. Les temps de calcul sont reportés en secondes et les longueurs de chemin en time-slots. A chaque fois, nous déterminons des intervalles de confiance à 95% des valeurs moyennes estimées afin d'évaluer le degré de validité statistique des comparaisons réalisées. Les intervalles de confiance sont calculés à partir de l'écart type obtenu pour 100 exécutions. Dans la plupart des tests réalisés, les algorithmes sont configurés en mode d'exécution court. Cela veut dire qu'une exécution est arrêtée dès lors qu'une solution admissible est rencontrée, ou sinon au bout d'une durée maximale allouée. Cette configuration, en exécution courte par défaut, se justifie par la demande des concepteurs et utilisateurs de la plateforme μSpider II. IlS sont beaucoup plus motivés par l'obtention d'une solution admissible que par la qualité de la solution. Mais, pour rester ouvert certains tests qui seront précisés sont réalisés en mode d'exécution long, avec une durée d'exécution longue imposée. Permettant ainsi d'avoir une meilleure qualité de la solution.

7.4 Conclusion

Nous avons présenté des jeux de tests structurés issus de cas réels d'application et un générateur de jeux de trafic aléatoires que nous avons utilisés pour évaluer nos méthodes de résolution. Dans le premier cas, des instances ont été conçues en fonction des besoins d'applications réelles et générées selon les principes du NoC μSpider II. Dans le deuxième cas, nous mettons l'accent sur la génération d'instances variées produites en très grand nombre, dans le but d'approfondir l'étude de la robustesse. Le générateur doit fournir de la diversité dans les instances générées, cela de manière rapide, sans nécessairement garantir l'optimalité du flot de données injecté dans le réseau. Nous verrons que pour un réseau de grande taille, il est aisé d'obtenir des instances aléatoires générées présentant un taux de saturation de trafic supérieur à celui des instances structurées fournies.

8 Évaluation des recherches locales

Dans ce chapitre, nous procédons à une évaluation comparative des trois versions de la méthode de recherche locale proposée. Nous évaluons tout d'abord l'impact des opérateurs de base. Cela est présenté dans la section 8.1. Ensuite, dans la section 8.2, nous évaluons leurs performances sur l'ensemble des instances des jeux de tests du problème standard CKPP, et rapportons les résultats en comparaison d'une résolution exacte du modèle linéaire en nombres entiers par un solveur standard. Nous approfondissons l'étude de la robustesse en utilisant le générateur d'instances aléatoires et étudions notamment l'impact du taux d'injection de trafic dans le réseau sur les performances de la résolution. Cela est détaillé dans la section 8.3. Nous terminons l'étude par une application au problème avec reconfiguration dynamique CRKPP qui met en lumière les limites de la recherche locale. Cela est présenté dans la section 8.4. Nous concluons ensuite le chapitre.

8.1 Impact des composants de base

8.1.1 Impact de la procédure d'amélioration

Ici, nous étudions les performances de l'opérateur de voisinage, encore appelé opérateur d'amélioration/réparation. Nous étudions son impact lorsqu'il est ajouté ou non après exécution de la construction parallèle gloutonne en tant que deuxième phase d'optimisation. L'algorithme considéré est la recherche aléatoire itérée (IRS) spécifiée par les algorithmes 6, 7 et 8 dans laquelle la procédure Dijkstra modifiée n'est pas utilisée et où seuls les nombres d'itérations des boucles de construction et d'amélioration sont modifiés. Plus précisément, nous étudions l'influence des opérations de construction et d'amélioration lorsque nous ajustons les paramètres *maxCount*, *maxConstruct*, et *maxImprove* de l'algorithme. L'exécution s'arrête dès lors qu'une solution admissible est trouvée, ou lorsque le nombre maximum d'itérations spécifié est atteint.

Les résultats sont présentés dans le tableau 8.1. Le test a été effectué sur l'instance N2(8-9-7-28) de la figure 7.2. A chaque fois, nous reportons des évaluations en moyenne sur 100 exécutions. Puisqu'il se peut qu'aucune solution admissible ne soit obtenue, nous donnons dans la ligne « chemins construits » , le pourcentage moyen de chemins

Table 8.1: Impact construction/amélioration

configuration itérations	Constr. 1	Constr. 1000	Constr. 100000	Constr./Imp. (100, 1000, 1000)	Constr./Imp. (200, 1000, 1000)	Constr./Imp. (300, 1000, 1000)
chemins construits (%)	52	91	93	99.4	99.9	100
durée (s)	0.005	0.05	4.8	3.89	4.53	4.12

construits relativement au nombre total de messages de l'instance. Nous donnons aussi la durée d'exécution moyenne en secondes dans la ligne « durée » . Les résultats numériques sont donnés dans six colonnes du tableau. Les trois premières colonnes correspondent à la construction seule, itérée respectivement 1, 1000 et 100000 fois. Les trois dernières colonnes correspondent à l'introduction de la boucle d'amélioration et à la réitération de la phase de construction/amélioration. Elles correspondent à une combinaison construction/amélioration avec $maxConstruct = 1000$ itérations internes de construction et $maxImprove = 1000$ itérations internes d'améliorations. La combinaison est itérée respectivement $MaxCount = 100$ fois, $MaxCount = 200$ fois et $MaxCount = 300$ fois. Le nombre d'itérations spécifié correspond à une limite maximum et l'algorithme s'arrête dès lors qu'une solution admissible est trouvée.

Nous pouvons constater que la construction seule ne permet pas de garantir 100 % de chemins construits malgré l'augmentation du nombre d'itérations et donc du temps de calcul jusqu'à plus de 4,8 secondes. En revanche, lorsque la procédure d'amélioration/réparation est ajoutée, nous pouvons observer une amélioration significative du nombre de chemins construits de 93% auparavant, à plus de 99%, ceci pour une durée d'exécution similaire d'environ 4 secondes de temps de calcul. Notons que 100% des chemins sont construits pour la dernière combinaison présentée sur la dernière colonne. Ceci illustre l'intérêt d'utiliser une procédure de réparation fondée sur un opérateur de voisinage ainsi que mis en œuvre dans la phase d'amélioration.

8.1.2 Impact de la procédure Dijkstra modifiée

La question se pose de savoir si la procédure Dijkstra modifiée définie par l'Algorithme 4, bien que gourmande en temps de calcul, est appropriée à un raffinement de la solution obtenue préalablement par la construction parallèle gloutonne. Nous étudions ici l'impact de la procédure Dijkstra modifiée sur la qualité de la solution, en l'utilisant ou non au sein de la recherche locale IRS. Alors que la construction parallèle gloutonne génère rapidement un grand nombre de chemins admissibles, la procédure Dijkstra modifiée, qui prend plus de temps, a pour rôle de calculer les chemins restants un à un. Pour mettre en évidence les performances de la procédure Dijkstra modifiée, nous utilisons les quatre instances basées sur les topologies N2 et N3 données par les figures 7.2-7.3. L'instance N3AB correspond au CRKPP, avec de la reconfiguration dynamique, et les autres instances au problème standard CKPP sans reconfiguration dynamique. Les évaluations sont réalisées sur la base de 100 exécutions par instance. La simu-

lation s'arrête une fois qu'une solution admissible est trouvée, ou quand une limite maximale de la durée d'exécution est atteinte. Cette limite est d'environ 30 secondes pour les instances du CKPP et d'environ 150 secondes pour l'instance du CRKPP. Seules les exécutions fournissant 100% de solutions admissibles sur les 100 exécutions sont reportées.

Table 8.2: Recherche locale avec/sans Dijkstra

Nx(T, N, NI, K)	IRS sans Dijkstra		IRS avec Dijkstra	
	durée(s)	longueur	durée(s)	longueur
N2(8-9-7-28)	4,12	121,04	0,11	119,5
N3A(9-15-10-27)	-	-	0,70	131,7
N3B(9-15-10-24)	-	-	0,82	119,6
N3AB(9-15-10-33)	-	-	-	-

Les résultats sont donnés dans la table 8.2. La première colonne indique le nom des instances avec leurs paramètres, tandis que les autres colonnes présentent les résultats pour les deux configurations de la recherche locale, respectivement sans ou avec la procédure Dijkstra modifiée. La première remarque à faire est la réduction importante du temps de calcul par l'ajout de la procédure Dijkstra modifiée sur l'instance N2. La recherche locale itérée sans la procédure Dijkstra modifiée ne permet pas de résoudre les instances N3A et N3B. L'ajout de la procédure Dijkstra permet de résoudre ces deux instances. La recherche locale fournit des solutions admissibles pour trois des instances considérées dans le temps imparti. Dans tous les cas de figures, nous constatons qu'elle ne fournit pas de solution satisfaisante pour le problème avec reconfiguration dynamique N3AB. Cela doit s'expliquer par la stagnation de la recherche dans des minima locaux et nous verrons que la diversité des solutions générées par l'approche évolutionnaire permettra de surmonter cette limite de la recherche locale.

8.2 Évaluation des recherches locales et méthode exacte

8.2.1 Application au problème standard

Dans cette section, nous comparons les trois méthodes de recherche locale appliquées sur les cinq instances structurées du problème standard CKPP. Nous rapportons également les résultats de la résolution exacte, obtenus avec un solveur standard appliqué au programme linéaire en nombres entiers (ILP) donné au chapitre 2. L'ILP a été résolu en utilisant la librairie (GLPK*).

Les trois méthodes de recherche locale sont implémentées par les Algorithmes 6 à 9 avec leurs paramètres par défaut. Dans la méthode de recherche aléatoire itérée (IRS), la combinaison des opérations de construction/amélioration est définie par les

valeurs $(maxCount, maxConstruct, maxImprove) = (50, 1000, 1000)$. Le paramètre $maxImprove$ est spécifique à la méthode IRS. Les deux autres méthodes de recherche locale considérées sont la recherche gloutonne ILS-FI et la recherche en profondeur ILS-BI. Ces deux méthodes consistent simplement à redéfinir la procédure d'amélioration selon le principe de l'Algorithme 9, dans lequel l'arrêt de l'amélioration par la règle de pivotement dépend de l'obtention d'un minimum local et non pas d'un nombre maximum d'itérations. Un paramètre important est la taille de l'échantillon examiné à chaque tentative de pivotement dans le voisinage. Ce paramètre est fixé à $neighborhood\text{-}SampleSize = 100$ pour ILS-FI et ILS-BI. Il ne doit pas être confondu avec la taille du voisinage, qui est déterminée par le nombre maximum de chemins pouvant être supprimés et reconstruits à chaque mouvement élémentaire dans le voisinage. Par défaut, nous optons pour un nombre de chemins supprimés aléatoire compris entre 1 et K. Ce paramètre correspond à un voisinage large. Nous étudions son impact dans la section suivante.

Les résultats numériques comparatifs sont présentés dans la table 8.3. La première colonne indique le nom de l'instance et ses paramètres caractéristiques (T, N, P, K), que sont respectivement la longueur du cycle d'émission, le nombre de routeurs, le nombre d'IP et le nombre de messages. Les 15 autres colonnes présentent les résultats pour les trois méthodes de recherche locale et deux configurations du solveur ILP. Pour chaque méthode, sont reportées la durée moyenne d'exécution, la longueur totale des chemins, et la longueur moyenne d'un chemin. Les résultats sont donnés en moyenne sur une base de 100 exécutions. Les instances sont classées par ordre de taille croissante. L'instance N1 est la plus petite, tandis que l'instance N4 est la plus grande. La dernière ligne donne les valeurs moyennes sur l'ensemble des instances avec un intervalle de confiance à 95% calculé en se basant sur les écarts types.

Nous pouvons observer dans la table 8.3 que les trois méthodes de recherche locale produisent des solutions dont les longueurs sont similaires, donc, de qualité similaire. Une amélioration légère du temps de calcul peut être observée pour les approches ILS-FI et ILS-BI, comparativement à IRS. La méthode IRS comporte peu de copies de données au regard des deux autres. Nous pensons qu'une implantation plus affinée de l'opérateur de voisinage dans les méthodes ILS-FI et ILS-BI, de manière à réduire les copies de données, est source potentielle d'amélioration de la performance, comme souvent avec les heuristiques standards[32]. Nous considérons deux configurations du *solver* du programme linéaire en nombres entiers dont les résultats sont reportés dans les 6 dernières colonnes de la table 8.3. Dans les colonnes « ILP-première-sol » , sont données les valeurs correspondantes à l'obtention de la première solution admissible rencontrée pendant une exécution. Cette solution n'est pas optimale. Les colonnes « ILP-opt-sol » donnent les valeurs trouvées pour la solution optimale. Seules les deux plus petites instances ont été résolues de manière exacte avec succès avec le solveur GLPK dans un temps de calcul raisonnable. Les autres instances n'ont pas donné de solution.

Table 8.3: Évaluation sur les instances structurées du CKPP

Nx(T, N, P, K)	IRS durée[a]	IRS long.	IRS long. moy.	ILS-FI durée[a]	ILS-FI long.	ILS-FI long. moy.	ILS-BI durée[b]	ILS-BI long.	ILS-BI long. moy
N1(6-7-4-12)	0,002	53,3	4,438	0,001	53,5	4,458	0,001	53,7	4,478
N2(8-9-7-28)	0,11	119,5	4,267	0,15	117,8	4,209	0,17	119	4,251
N3A(9-15-10-27)	0,70	131,7	4,877	0,69	129,8	4,809	0,66	127,1	4,708
N3B(9-15-10-24)	0,82	119,6	4,982	0,84	118,6	4,942	0,85	118	4,915
N4(47-36-35-209)	8,89	1565	7,488	7,25	1570	7,511	7,57	1568	7,501
moyenne	2,10($\pm 0,14$)	398($\pm 3,52$)	5,210($\pm 0,031$)	1,79($\pm 0,1$)	398($\pm 3,5$)	5,186($\pm 0,030$)	1,85($\pm 0,13$)	397(± 3)	5,171($\pm 0,028$)

Nx(T, N, P, K)	ILP-première-sol durée[a]	ILP-première-sol long.	ILP-première-sol long. moy.	ILP-opt-sol durée[a]	ILP-opt-sol long.	ILP-opt-sol long. moy.
N1(6-7-4-12)	21	45	3,75	720	42	3,5
N2(8-9-7-28)	185	140	5	8400	76	2,71
N3A(9-15-10-27)	-	-	-	-	-	-
N3B(9-15-10-24)	-	-	-	-	-	-
N4(47-36-35-209)	-	-	-	-	-	-
moyenne	-	-	-	-	-	-

[a] Temps d'exécution en secondes avec Intel Core Duo (3.0 GHz) (1 cœur utilisé), programme C++.
[b] Temps d'exécution en secondes avec Intel Celeron 585 (2.16 GHz) secondes, *solveur* GLPK.

8.2.2 Impact des tailles de voisinage

Nous proposons de mieux évaluer l'impact de la taille du voisinage sur les performances. La taille du voisinage est le nombre maximum de chemins supprimés/reconstruits à chaque mouvement. Les tests de la section précédente sont basés sur un voisinage large, avec un nombre de chemins supprimés/reconstruits tiré aléatoirement à chaque mouvement et compris entre 1 et K. Nous évaluons la performance suivant une diminution progressive de la taille du voisinage. Nous considérons un voisinage moyen en prenant un nombre maximum de chemins supprimés entre 1 et $K/2$, et un petit voisinage si nous restreignons le mouvement à exactement 2 messages supprimés/reconstruits.

Nous évaluons chaque stratégie de recherche locale (IRS, ILS-FI, ILS-BI) pour chaque taille du voisinage (large, moyen, petit) sur l'ensemble des instances N1, N2, N3A, N3B et N4. Chaque exécution est réitérée 100 fois et les résultats considérés en moyenne. Les résultats sont présentés dans la figure 8.1 avec un intervalle de confiance à 95%. Les deux graphiques de la figure nous permettent d'évaluer le compromis entre qualité de la solution (longueur) et temps de calcul. Le graphique de gauche montre que les longueurs obtenues sont plutôt similaires. Comme tous les intervalles de confiance se chevauchent, les différences de longueurs ne semblent pas statistiquement significatives. Certaines disparités apparaissent entre les temps de calcul, comme le montre le graphique de droite de la figure.

Le premier point à noter est la similitude de comportement entre les méthodes ILS-BI et ILS-FI avec la diminution de la taille du voisinage. Ceci n'est pas surprenant puisque les deux méthodes sont similaires et examinent un échantillon relativement

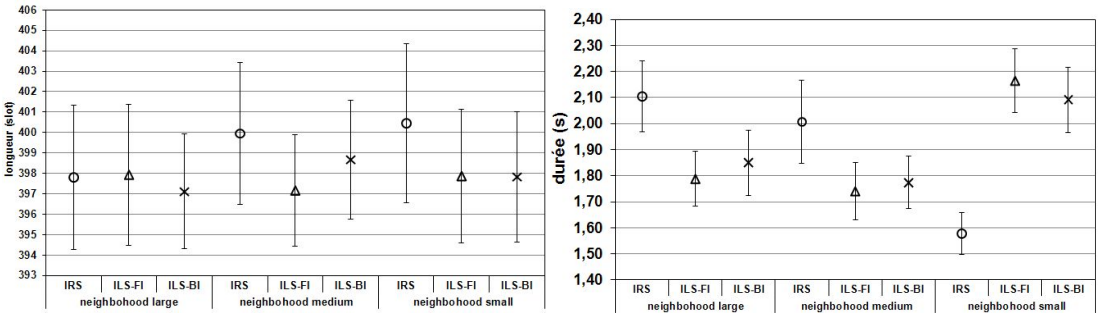


Figure 8.1: Impact de la taille du voisinage sur les longueurs et les durées d'exécution.

petit de solutions dans le voisinage. Le second point à noter est le contraste de comportement entre les méthodes ILS d'une part et IRS d'autre part. Alors que les premières sont sensiblement moins rapides avec un petit voisinage, la situation est inversée pour la seconde. La méthode IRS est plus rapide avec un petit voisinage. Cela met en lumière le compromis entre intensification et diversification de la recherche dans les deux méthodes. Les ILS cherchent une amélioration autour d'un élément pivot, dans une petite zone de recherche. L'extension de la taille du voisinage augmente alors la diversification. Au contraire, IRS effectue sa recherche dans des directions aléatoires par transformation de la solution précédente. Cela favorise naturellement la diversification. Diminuer la taille du voisinage réduit la zone de recherche favorablement. En considérant le compromis entre temps de calcul et longueur, nous utiliserons par la suite ou bien la méthode ILS-FI avec un voisinage large, lorsque nous cherchons un gain en qualité, ou bien la méthode IRS avec un petit voisinage, lorsque nous cherchons un gain en durée d'exécution.

8.3 Évaluation sur des jeux de tests aléatoires

8.3.1 Application au problème standard

Le générateur d'instances aléatoires que nous avons développé et donné par l'Algorithme 13, nous a permis de compléter l'analyse des performances des recherches locales. A partir des cinq instances structurées fournies par R. Dafali et de leur graphe de communication respectifs, nous générons, pour chaque instance structurée, 100 nouvelles instances de trafic aléatoires avec un seuil minimum de deux paquets par message. Nous appliquons l'algorithme de recherche locale sur ces 100 instances et évaluons les résultats moyens. Seule la recherche locale ILS-FI est utilisée.

Les résultats sont présentés dans la table 8.4. Les deux colonnes « TSL-réel » et « TSL-aléatoire » donnent respectivement le niveau de saturation de trafic de l'instance

structurée de départ et celui en moyenne des 100 jeux de trafic générés aléatoirement. La colonne « TSL max » donne le niveau de saturation le plus élevé parmi les 100 instances générées aléatoirement. Les résultats de la méthode ILS-FI sont exprimés par la durée d'exécution en secondes, la longueur totale et moyenne des chemins, ainsi que par le pourcentage de chemins construits et le pourcentage de solutions admissibles trouvées parmi les 100 exécutions. La dernière ligne du tableau donne les valeurs moyennes pour l'ensemble des jeux de tests.

D'une part, à l'exception de l'instance N4, toutes les instances sont résolues avec succès, et avec un temps de calcul similaire ou plus court que pour les instances structurées de départ, si l'on se reporte également au tableau 8.3. On peut remarquer qu'à l'exception de l'instance N4, les TSL des jeux de trafic aléatoires sont tous semblables ou inférieurs à ceux des instances structurées de départ. D'autre part, l'instance N4 n'est jamais résolue avec succès, puisqu'en moyenne seuls 94% des chemins sont construits, et aucune solution admissible n'est obtenue. Cela peut s'expliquer par la valeur du TSL élevé pour les jeux de test aléatoires de type N4 qui est de 39,97% en moyenne. Ce TSL est 50% plus élevé que celui de l'instance structurée de départ à 26%. Ce paramètre ayant un impact sur l'effectivité de la résolution, nous allons examiner plus finement son influence dans le cas du test N4 aléatoire.

Table 8.4: Évaluation sur les instances aléatoires du CKPP avec ILS-FI

Nx(T, N, P, K)	TSL-réel(%)	TSL-aléatoire(%)	TSL max(%)	IlS-FI durée(s)	long.	long. moy.	constr.(%)	admis.(%)
N1-aléatoire(6-7-4-12)	100	100	100	0,002	53	4,418	100	100
N2-aléatoire(8-9-7-28)	100	100	100	0,17	117,5	4,197	100	100
N3A-aléatoire(9-15-10-27)	90	87,86	90	0,49	126,9	4,701	100	100
N3B-aléatoire(9-15-10-24)	90	85,54	89	0,50	117	4,877	100	100
N4-aléatoire(47-36-35-209)	26,02	39,97	40	618,45	1405	7,162	93,86	0
moyenne	81,20	82,67($\pm 0,1$)	83	123,92($\pm 0,8$)	363,9($\pm 2,4$)	5,071($\pm 0,028$)	-	-

8.3.2 Impact du taux de saturation de trafic

Pour évaluer à partir de quel niveau de trafic injecté l'algorithme devient efficace, nous procédons à de nouvelles expérimentations en faisant varier le TSL. Pour cela, nous supprimons progressivement des paquets des messages selon une succession d'instances de TSL décroissant. Nous obtenons 100 instances aléatoires par degré de TSL examiné. Les résultats d'exécution sont donnés par les deux graphiques de la figure 8.2. Sur celui de gauche, sont respectivement représentés le pourcentage de chemins construits et celui de solutions admissibles trouvées en fonction du TSL. Sur celui de droite, sont donnés les temps d'exécution en fonction du TSL. Nous pouvons observer qu'une construction de 100% des chemins n'est obtenue qu'en dessous d'un niveau intermédiaire de TSL d'environ 30%. Cela signifie que de 26% initialement, le TSL peut-être augmenté jusqu'à 30% tout en préservant la topologie du NoC structuré N4. Comme dans un problème de *bin packing*, des paquets sont injectés dans les liens de transmission

jusqu'à l'apparition d'une surcharge. A notre avis, cette expérience met en lumière la difficulté de dimensionnement du réseau, lorsque que le concepteur doit maximiser les quantités de paquets envoyés, tout en garantissant l'acheminement sans contention dans une topologie particulière, cela particulièrement pour des NoC de grande taille.

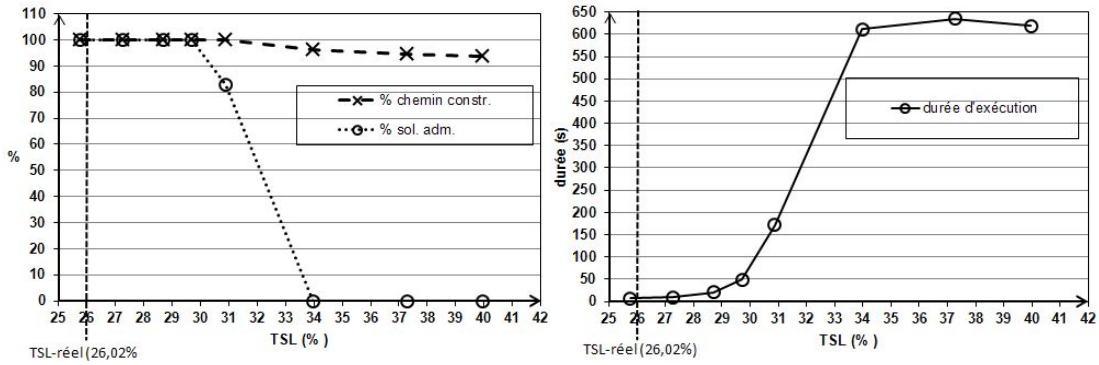


Figure 8.2: Performance selon le taux d'injection de trafic dans le NoC N4.

8.4 Application au problème avec reconfiguration dynamique

Une limitation importante des approches de recherche locale est qu'elles ne permettent pas dans le cas présent de résoudre la version du problème de routage avec reconfiguration dynamique CRKPP. Nous rapportons dans le tableau 8.5, le détail des résultats obtenus avec la méthode IRS (la plus rapide) appliquée à l'instance N3AB respectivement en exécution courte et longue. Dans le premier cas, 3 solutions admissibles sont construites sur 100 exécutions, pour 40 dans le second cas, dans le temps imparti. Les phases construction/amélioration sont réitérées 50 fois ($maxCount = 50$) en exécution courte, et 250 fois ($maxCount = 250$) en exécution longue.

Table 8.5: Recherche locale sur l'instance structurée du CRKPP

	IRS exécution courte				
Nx(T, N, P, K)	durée(s)	long.	long. moy.	constr.(%)	admis.(%)
N3AB(9-15-10-33)	23,77	143,41	4,519	96,12	3
moyenne	23,77($\pm 0,45$)	143,41($\pm 1,74$)	4,519($\pm 0,046$)	-	-

	IRS exécution longue				
Nx(T, N, P, K)	durée(s)	long.	long. moy.	constr.(%)	admis.(%)
N3AB(9-15-10-33)	109	148,5	4,583	98,2	40
moyenne	109($\pm 9,9$)	148,5($\pm 1,89$)	4,583($\pm 0,058$)	-	-

8.5 Conclusion

Nous avons procédé aux évaluations des trois versions de la méthode de recherche locale, en essayant de mettre en évidence la complémentarité des composants algorithmiques de base, et en comparant les performances suivant leurs différentes versions et paramètres d'exécution. Ainsi, l'influence de la taille du voisinage a été considérée ainsi que celle du niveau de saturation en trafic du réseau. Il apparaît que l'ajustement des quantités de paquets selon la dimension du réseau est une tâche délicate qui incombe au concepteur du NoC. Des tentatives de résolution exacte du programme linéaire en nombres entiers n'ont abouties que pour les deux plus petites instances. Une limite de la recherche locale apparaît lorsque nous cherchons à résoudre le problème de routage avec reconfiguration dynamique, manifestement plus difficile à résoudre que la version standard. Pour surmonter cette limite, nous proposons de modifier notre stratégie de recherche en adoptant une approche à base de population de solutions de type algorithme évolutionnaire. Il s'agit de favoriser la diversification de la recherche et d'échapper plus aisément aux minima locaux. C'est l'objet du prochain chapitre.

9 Évaluation de l'approche évolutionnaire

Nous mettons l'accent dans ce chapitre sur l'évaluation de l'algorithme évolutionnaire, tout en procédant à une étude comparée des différentes approches que nous avons développées, pour résoudre les instances structurées et aléatoires du CKPP et du CRKPP. De même que nous l'avons fait pour la recherche locale, nous commençons par évaluer l'impact de la procédure Dijkstra modifiée au sein de l'algorithme évolutionnaire, dans la section 9.1. Nous abordons ensuite l'évaluation des différentes stratégies de recherche sur les jeux de test du problème standard et ensuite sur ceux du problème avec reconfiguration dynamique, respectivement dans les sections 9.2 et 9.3. Dans ces sections, on montre les différents compromis obtenus et propriétés relatives des approches sur le problème. Seules les algorithmes évolutionnaire et algorithme mémétique permettent une résolution efficace du problème avec reconfiguration dynamique. Nous terminons les expérimentations par la prise en compte des versions min-sum et min-max du problème, en examinant l'effet d'un simple changement de fonction objectif au sein des algorithmes. Cela est présenté dans la section 9.4. Une conclusion termine le chapitre.

9.1 Impact de la procédure Dijkstra modifiée

Ici, nous étudions la différence de performances de l'algorithme évolutionnaire avec ou sans la procédure Dijkstra modifiée de l'Algorithme 4. Avec comme finalité l'obtention de la première solution admissible et non la recherche de la meilleure solution. Ce critère d'arrêt étant justifié par une demande des électroniciens de la plateforme μSpider II dont nous tenons compte. Par contre, à la section nous procédons à des expérimentations ayant pour objectif la recherche de la meilleure solution et non celle de la première solution admissible pour être en parfait accord avec un des principe de l'optimisation. Le tableau 9.1 illustre les différences de performances, dans l'obtention de la première solution admissible, sur les instances N2, N3A, N3B et N3AB. Les résultats sont donnés en moyenne sur 100 exécutions. La première constatation est l'amélioration très nette des performances avec l'utilisation du Dijkstra modifié. Les tests ont été réalisés ici avec une limite maximum de temps d'exécution de 30 secondes pour les jeux de test N3A et N3B et de 150 secondes pour le test N3AB. Les résultats ne sont rapportés que dans le cas où l'ensemble des 100 exécutions ont produit chacune

une solution admissible dans la limite de temps.

Table 9.1: EA avec/sans Dijkstra

Nx(T, N, NI, K)	EA sans Dijkstra durée(s)	EA sans Dijkstra longueur	EA avec Dijkstra durée(s)	EA avec Dijkstra longueur
N2(8-9-7-28)	2,7	120,72	0,14	126,9
N3A(9-15-10-27)	6,84	122,27	0,47	132,5
N3B(9-15-10-24)	7,15	115,72	0,38	123
N3AB(9-15-10-33)	-	-	11,05	159,95

L'algorithme évolutionnaire permet de résoudre les trois instances du CKPP, avec ou sans la procédure Dijkstra. Les temps d'exécution moyens de l'algorithme évolutionnaire pour obtenir une solution admissible sont divisés par d'environ 10 par rapport au cas sans utilisation du Dijkstra modifié, pour une longueur totale moyenne des chemins à peu près équivalente. Le bénéfice de l'approche à base de population apparaît surtout lorsqu'elle est appliquée à l'instance N3AB du problème de routage avec reconfiguration dynamique des chemins (CRKPP). L'algorithme évolutionnaire incluant la procédure Dijkstra réussit à résoudre ce cas dans le délai imparti. Nous soutenons que l'instance CRKPP est considérablement plus difficile à résoudre que les instances CKPP, puisque la résolution de l'instance N3AB implique la résolution de 2^3 instances du CKPP (incluant N3A et N3B), de telle sorte que chaque solution du CKPP est compatible avec tous les interchangements possibles de tables TDMA sur début de cycle d'émission lors de l'exécution du NoC. Les performances atteintes par l'algorithme évolutionnaire peuvent s'expliquer par la hausse du niveau de diversification atteint, qui se produit avec une recherche basée sur une population de solutions.

9.2 Application au problème standard

Dans cette section, nous évaluons l'approche évolutionnaire et mémétique sur le problème standard.

9.2.1 Résultats sur les jeux de tests structurés

Nous comparons les différentes méthodes sur les cinq instances structurées relevant du CKPP. Cette comparaison comprend l'algorithme évolutionnaire (EA), l'algorithme mémétique (MA) et la recherche locale IRS. Pour les approches à base de population, la taille de celle-ci est fixée à 100 individus. C'est à partir d'expérimentations préalables, non répertoriées ici, que nous avons pu déduire un compromis entre la taille 100 de la population et la diversité des individus de la population. La méthode de recherche locale qui est inclue dans l'algorithme mémétique est la méthode IRS avec un petit voisinage. Du fait du nombre d'itérations fixé au sein de IRS, cela confère une régularité

d'exécution à l'algorithme mémétique. Des tests préliminaires indiquent que seule cette configuration de recherche locale confère à l'algorithme mémétique de bonnes performances, les autres recherches locales entraînant des temps d'exécution trop élevés difficiles à prévoir. De même que dans tous les tests précédents, nous évaluons les algorithmes selon leur capacité à produire une solution admissible le plus vite possible. Pour rappel, une solution consiste à déterminer l'instant d'émission de chaque message, et les chemins origine/destination sans conflit de l'ensemble des messages. Une solution ainsi obtenue est dite admissible. Sinon, elle est dite non admissible si au moins une des messages à son chemin non construit. L'exécution est stoppée dès lors qu'une solution admissible est trouvée, ou dans le cas contraire lorsqu'une limite maximum de temps est atteinte. Le nombre maximum de générations est de 60000 pour EA et de 12000 pour MA. Ces nombres ayant été fixés pour avoir suffisamment de temps d'exécution.

Les résultats numériques comparatifs sont présentés dans la table 9.2. Ils correspondent à la méthode IRS à petit voisinage, à EA et à MA. Excepté sur N4, l'algorithme évolutionnaire semble aussi performant que IRS. L'algorithme mémétique en revanche est nettement plus lent sur N3A et N3B. Sur N4, la situation est différente puisque l'algorithme évolutionnaire est bien plus lent que IRS et MA. Les figures 9.1 et 9.2 permettent d'illustrer la situation respectivement sur les cas N3A et N4. L'algorithme évolutionnaire (EA) est moins performant que IRS et EA dans cas N4, sans doute en raison de sa grande taille. Du fait d'une population de 100 individus, les opérations élémentaires de l'algorithme évolutionnaire s'appliquent désormais avec une lenteur accrue. L'algorithme mémétique hérite en revanche des propriétés de la recherche locale qui est inclue en tant qu'opérateur. Il se peut que très peu d'appels de cette recherche locale suffisent à générer une solution admissible, la dynamique de population n'ayant en fait pas un grand impact dans ce cas précis.

Table 9.2: Évaluation sur les instances structurées du CKPP

	IRS			EA			MA		
Nx(T, N, P, K)	durée[a]	long.	long. moy.	durée[a]	long.	long. moy.	durée[a]	long.	long. moy.
N1(6-7-4-12)	0,002	52,8	4,400	0,024	47,6	3,970	0,006	53,5	4,460
N2(8-9-7-28)	0,17	118,9	4,247	0,14	126,9	4,531	0,20	120,4	4,301
N3A(9-15-10-27)	0,61	132,9	4,924	0,47	132,5	4,907	6,25	131,2	4,857
N3B(9-15-10-24)	0,61	120,3	5,011	0,38	123	5,124	8,48	119,7	4,988
N4(47-36-35-209)	6,50	1577	7,547	66,17	1761	8,425	19,78	1566	7,493
moyenne	1,58($\pm 0,08$)	400($\pm 3,9$)	5,226($\pm 0,031$)	13,44($\pm 0,9$)	438($\pm 2,3$)	5,391($\pm 0,025$)	6,94($\pm 0,8$)	398($\pm 2,9$)	5,220($\pm 0,030$)

[a] Temps d'exécution en secondes avec Intel Core Duo (3.0 GHz) (1 cœur utilisé), programme C++.

9.2.2 Résultats sur des jeux de trafic aléatoires

Nous procédons ici à une comparaison de l'algorithme évolutionnaire avec la recherche locale ILS-FI sur des jeux de trafic aléatoires. Il s'agit de vérifier la robustesse de l'algorithme évolutionnaire sur des jeux de données variés. Pour chaque instance structurée de départ, sont générées 100 nouvelles instances de trafic avec un seuil mini-

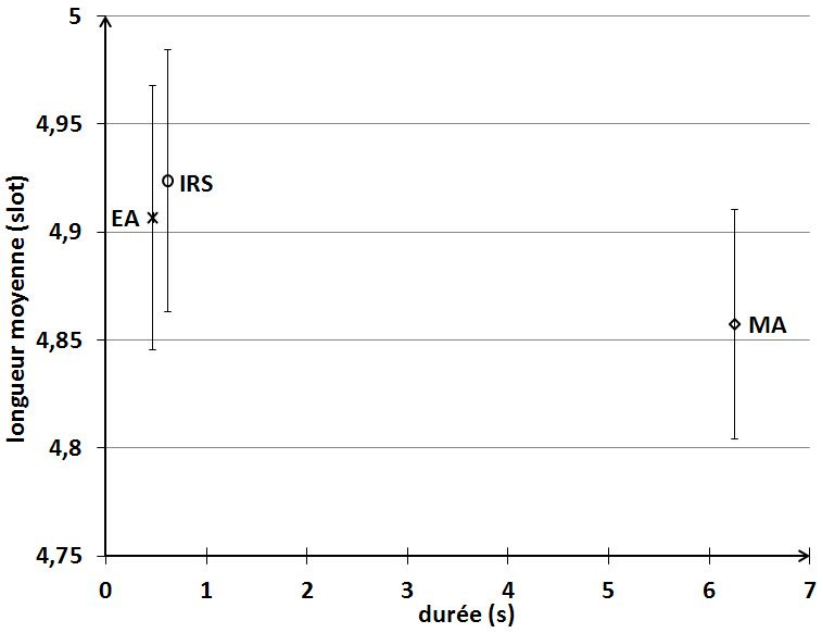


Figure 9.1: IRS, EA, MA sur N3A.

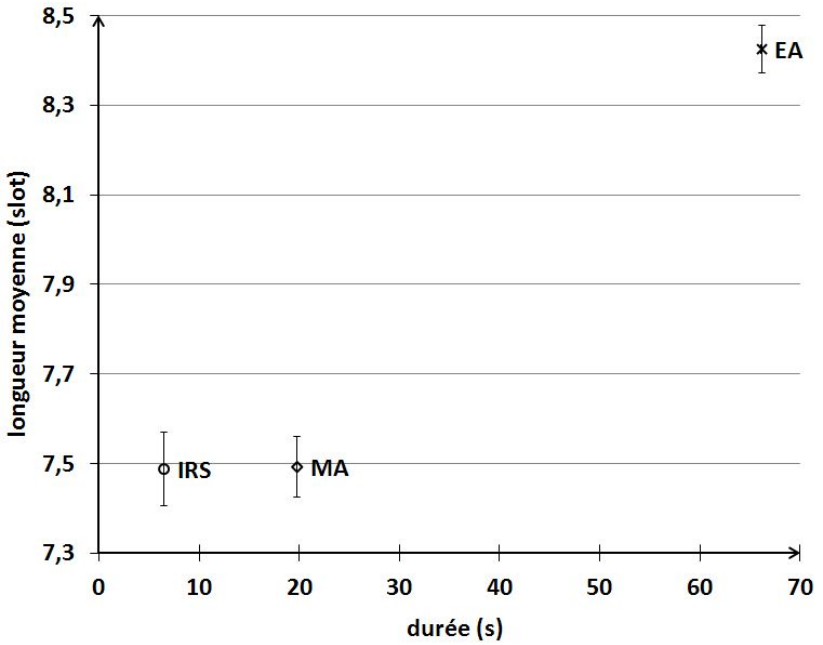


Figure 9.2: IRS, EA, MA sur N4.

mum de deux paquets par message. Nous appliquons les algorithmes de recherche et évaluons les valeurs moyennes sur les 100 exécutions. Les résultats numériques sont consignés dans la table 9.3. Les deux colonnes « TSL-réel » et « TSL-aléatoire » donnent respectivement le niveau de saturation de trafic des instances structurées de départ et une moyenne de ceux générés aléatoirement. La colonne « TSL max » reporte le niveau de saturation le plus élevé sur les 100 instances générées aléatoirement. Le premier constat est que ILS-FI et EA présentent globalement un comportement similaire sur ces instances. Le deuxième constat est que ni ILS-FI, ni EA n'arrivent à résoudre N4, la plus grande des instances dans le cas aléatoire. De même que dans l'analyse de la section 8.3 du chapitre 8, nous l'expliquons par le niveau de saturation de trafic élevé de l'instance aléatoire qui se situe au-dessus du seuil d'efficacité des deux méthodes. Nous renvoyons à la section en question pour plus de détails sur l'influence du taux de saturation de trafic lors de la résolution.

Table 9.3: Évaluation sur des instances aléatoires du CKPP

	TSL			IlS-FI				
Nx(T, N, P, K)	réel(%)	aléa.(%)	max (%)	durée(s)	long.	long. moy.	constr.(%)	admis.(%)
N1-aléatoire(6-7-4-12)	100	100	100	0,002	53	4,418	100	100
N2-aléatoire(8-9-7-28)	100	100	100	0,17	117,5	4,197	100	100
N3A-aléatoire(9-15-10-27)	90	87,86	90	0,49	126,9	4,701	100	100
N3B-aléatoire(9-15-10-24)	90	85,54	89	0,50	117	4,877	100	100
N4-aléatoire(47-36-35-209)	26,02	39,97	40	618,45	1405	7,162	93,86	0
moyenne	81,20	82,67($\pm 0,1$)	83	123,92($\pm 0,8$)	363,9($\pm 2,4$)	5,071($\pm 0,028$)	-	-

	TSL			EA				
Nx(T, N, P, K)	réel(%)	aléa.(%)	max (%)	durée(s)	long.	long. moy.	constr.(%)	admis.(%)
N1-aléatoire(6-7-4-12)	100	100	100	0,03	47,9	3,988	100	100
N2-aléatoire(8-9-7-28)	100	100	100	0,14	127,2	4,543	100	100
N3A-aléatoire(9-15-10-27)	90	87,86	90	0,30	132,7	4,915	100	100
N3B-aléatoire(9-15-10-24)	90	85,54	89	0,37	121,8	5,074	100	100
N4-aléatoire(47-36-35-209)	26,02	39,97	40	389	1432	7,373	92,94	0
moyenne	81,20	82,67($\pm 0,1$)	83	77,78($\pm 0,9$)	372($\pm 2,2$)	5,179($\pm 0,027$)	-	-

9.3 Application au problème avec reconfiguration dynamique

L'intérêt de l'approche évolutionnaire réside dans son potentiel à traiter le problème avec reconfiguration dynamique CRKPP. Ici, nous comparons les méthodes IRS, EA et MA sur ce problème en utilisant l'instance structurée N3AB et ensuite ses jeux de trafic aléatoires dérivés. Nous rapportons la durée d'exécution, la longueur totale et moyenne des chemins construits, la proportion de chemins effectivement construits et la proportion de solutions admissibles obtenues sur les 100 exécutions réalisées. La méthode de recherche locale inclue dans l'algorithme mémétique est la méthode IRS avec petit voisinage. La simulation s'arrête dès lors qu'une solution admissible est trouvée, ou lorsqu'une limite maximale de durée d'exécution est atteinte.

9.3.1 Résultats sur les jeux de tests structurés

L'instance considérée pour ces tests est celle fournie par R. Dafali comme provenant d'une application réelle. Ainsi que nous l'avons déjà mentionné, cette instance présente un TSL de 90% pour un MRT de 100%. Pour rappel, le TSL (*Trafic Saturation Level*) est défini par l'équation (3.6) du chapitre 3. Il reflète le niveau maximum de trafic injecté dans le réseau à un moment donné. Le MRT (*Maximum Reception Throughput*) est défini par l'équation (3.7) du chapitre 3, il exprime le taux maximum d'absorption de paquets par un récepteur, considéré à différents moments suivant l'application des différentes configurations de tables TDMA. Un MRT de 100% signifie que tout récepteur peut-être utilisé à 100% de sa capacité d'absorption de trafic avec une configuration dynamique de tables TDMA appropriée.

Des résultats numériques de la table 9.4, il ressort que les approches à base de population permettent de résoudre le problème reconfigurable, tandis que cela n'est pas le cas pour la recherche locale comme présenté avec la méthode IRS. Il apparaît que EA est de loin la solution la plus efficace pour résoudre l'instance N3AB. Cela est certainement dû à la grande diversité de solutions générées au sein de la population de solutions et continuellement modifiées par les opérateurs élémentaires de manipulation de chemins. L'algorithme mémétique bénéficie aussi de la diversité présente dans la population de solutions, mais présente une grande lenteur d'exécution du fait de la répétition d'appels aux recherches locales internes, chacune étant unitairement coûteuse en temps de calcul. La recherche locale seule, bien que réitérée sur une durée maximum d'environ 150 secondes, ne fournit que 40% de solutions admissibles. Dans le cas présent, la combinaison des opérations de construction/ amélioration est définie par les valeurs $(maxCount, maxConstruct, maxImprove) = (250, 1000, 1000)$.

Les deux figures 9.3 et 9.4 permettent de résumer les propriétés respectives des trois méthodes IRS, EA, et MA suivant leur application aux cas de test N3AB et N4. Le premier test N3AB représente la possibilité de résolution du problème avec reconfiguration dynamique. Le deuxième test N4 représente une instance de grande taille du problème standard. On remarquera les compromis réalisés par les différentes approches en termes de qualité de solution et durée d'exécution, d'une part, et suivant le type d'instance N3AB et N4, d'autre part. Si l'approche mémétique MA permet de surmonter les limites de la recherche locale, elle n'en reste pas moins nettement moins performante que l'approche EA sur le cas N3AB. Elle est cependant plus rapide sur N4, puisqu'elle reproduit davantage les performances de IRS sur ce cas précis. L'approche MA permet en partie de réutiliser les propriétés des deux types d'approches IRS et EA, au sein d'une même méthode de recherche.

9.3.2 Résultats sur des jeux de trafic aléatoires

Il s'agit de vérifier la robustesse de l'algorithme évolutionnaire sur des jeux de données variés du problème avec reconfiguration dynamique. Nous procédons ici à une comparaison de l'algorithme évolutionnaire AE avec la recherche locale ILS-FI sur 100 nou-

Table 9.4: Évaluation sur l'instance structurée du CRKPP

Nx(T, N, P, K)	IRS durée(s)	IRS long.	IRS long. moy.	IRS constr.(%)	IRS admis.(%)	EA durée(s)	EA long.	EA long. moy.	EA constr.(%)	EA admis.(%)
N3AB(9-15-10-33)	109	148,5	4,583	98,2	40	11,05	153,95	4,665	100	100
moyenne	109($\pm 9,9$)	148,5($\pm 1,89$)	4,583($\pm 0,058$)	-	-	11,05($\pm 3,4$)	153,95($\pm 1,45$)	4,665($\pm 0,044$)	-	-

Nx(T, N, P, K)	MA durée(s)	MA long.	MA long. moy.	MA constr.(%)	MA admis.(%)
N3AB(9-15-10-33)	103,10	158,98	4,818	100	100
moyenne	103,10(± 11)	158,98($\pm 1,3$)	4,818($\pm 0,04$)	-	-

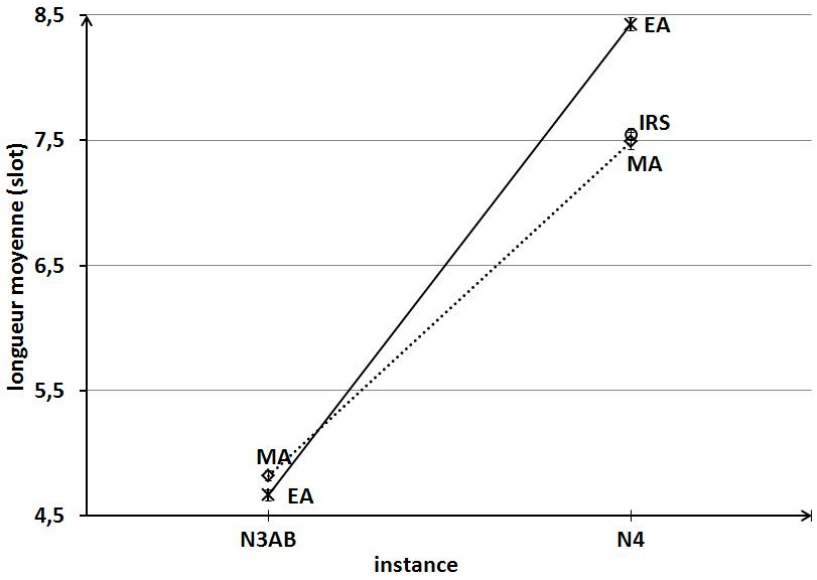


Figure 9.3: Comparaison en qualité de IRS, EA et MA sur N3AB et N4.

velles instances de trafic obtenues à partir du jeu de test N3AB avec un seuil minimum de deux paquets par message. De même que précédemment, nous appliquons les algorithmes de recherche et évaluons les valeurs moyennes sur les 100 exécutions. Les résultats numériques sont présentés dans la table 9.5. Les deux colonnes « MRT-réel » et « MRT-aléatoire » donnent le taux maximum de saturation d'un récepteur, ainsi que défini par l'équation (3.7) du chapitre 3, respectivement de l'instance structurée de départ et en moyenne sur les 100 jeux de trafic générés. La colonne « MRT max » reporte le MRT le plus élevé sur les 100 instances générées aléatoirement.

Le premier constat en examinant les résultats de la table 9.5, est que la recherche locale ILS-FI ne parvient pas à résoudre l'intégralité des instances reconfigurables aléatoires. Cela ne contredit pas le résultat de la section précédente où la méthode IRS échoue sur le cas structuré de départ. Le deuxième constat est que l'algorithme évolu-

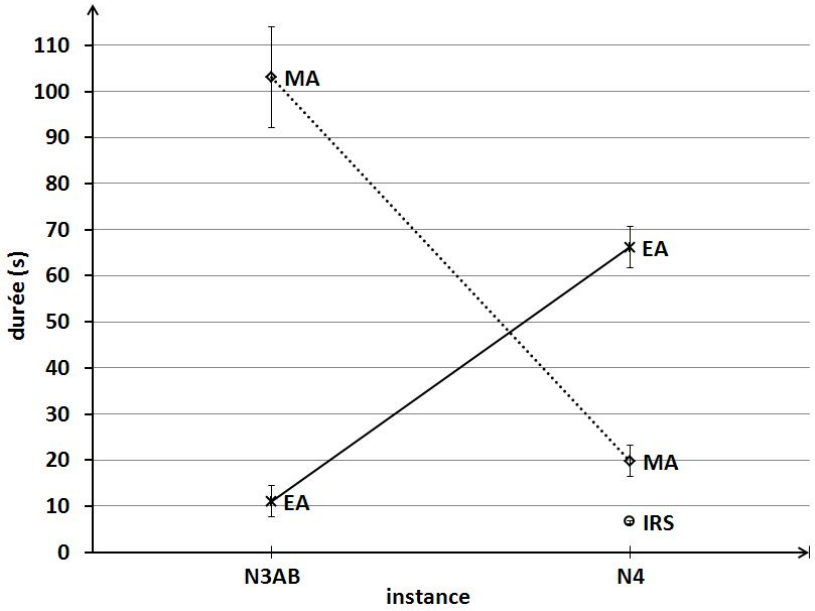


Figure 9.4: Comparaison en durée de IRS, EA et MA sur N3AB et N4.

tionnaire EA résout ces instances aléatoires sans plus de difficulté qu'il résout l'instance structurée de départ. On peut remarquer que les MRT générés aléatoirement sont légèrement inférieurs à 100%, alors que le MRT de l'instance de départ est de 100 % exactement. Cela explique le léger gain en durée de calcul observé ici pour EA par rapport au jeu structuré de départ. De même, cela semble indiquer que le concepteur peut, dans le cas d'un jeu de test de cette dimension, spécifier assez aisément une utilisation reconfigurable du NoC exploitant au mieux la capacité des récepteurs à différents moments.

Table 9.5: Évaluation sur des instances aléatoires du CRKPP

	MRT			ILS-FI				
Nx(T, N, P, K)	réel(%)	aléa.(%)	max (%)	durée(s)	long.	long. moy.	constr.(%)	admis.(%)
N3AB-aléatoire(9-15-10-33)	100	94,03	97,78	17,20	145,92	4,508	98,03	37
moyenne	-	94,03($\pm 0,09$)	-	17,20($\pm 1,57$)	145,92(± 2)	4,508($\pm 0,052$)	-	-

	MRT			EA				
Nx(T, N, P, K)	réel(%)	aléa.(%)	max (%)	durée(s)	long.	long. moy.	constr.(%)	admis.(%)
N3AB-aléatoire(9-15-10-33)	100	94,03	97,78	9,40	152,19	4,612	100	100
moyenne	-	94,03($\pm 0,09$)	-	9,40(± 3)	152,19($\pm 1,47$)	4,612($\pm 0,044$)	-	-

9.4 Versions min-sum et min-max du problème

Nous terminons nos évaluations expérimentales par la prise en compte simultanée des versions min-sum et min-max du problème de routage. Il s'agit d'évaluer la différence de performance des algorithmes suivant que l'on cherche à minimiser respectivement la longueur moyenne d'un chemin ou la longueur maximum d'un chemin. Pour cela, les algorithmes sont implémentés avec deux versions différentes de la fonction objectif. Dès lors qu'une évaluation ou comparaison de solutions est requise au sein des algorithmes, ou bien la longueur totale est considérée (fonction objectif min-sum), ou bien la longueur du chemin le plus long est considérée (fonction objectif min-max). Ici, nous limitons l'étude au problème standard représenté par l'instance de plus grande taille N4.

Les résultats numériques obtenus par les approches ILS-FI et EA sont donnés dans les tables 9.6 et 9.7, respectivement pour une exécution courte et une exécution longue. Dans le premier cas, l'algorithme s'arrête dès lors qu'une solution admissible est trouvée, comme dans la plupart des tests présentés jusqu'ici. Dans le deuxième cas, l'algorithme s'arrête au bout d'un nombre d'itérations relativement large correspondant à une durée d'exécution longue. Le contenu des tableaux est résumé dans les figures 9.5 et 9.6. Globalement, en mode d'exécution court, la modification de fonction objectif ne semble pas avoir d'impact significatif sur les performances en termes de longueur maximum ou moyenne. En effet, dans ce cas, l'algorithme s'arrête dès lors qu'une solution admissible est rencontrée, ce qui ne laisse pas suffisamment de temps à ce mécanisme de produire un effet particulier sur les longueurs maximum ou moyenne des chemins. En revanche, l'intérêt de ce mécanisme simple apparaît en mode d'exécution long. Nous pouvons constater, dans ce cas, une corrélation entre la configuration de l'algorithme et sa capacité à minimiser l'objectif requis correspondant. Par exemple, la longueur maximum obtenue est nettement inférieure pour une configuration min-max de l'algorithme, tandis que la longueur moyenne obtenue est nettement inférieure pour une configuration min-sum de l'algorithme.

Table 9.6: Résultats en exécution courte

	ILS-FI				EA			
Version	durée(s)	long.	long. moy.	long. max chemin	durée(s)	long.	long. moy.	long. max chemin
min-sum	6,92	1566	7,494	59,89	64,19	1765	8,443	42,18
min-max	7,21	1563	7,480	61,89	78,51	1776	8,495	41,42
moyenne	7,1($\pm 0,14$)	1565($\pm 4,66$)	7,487($\pm 0,022$)	60,89($\pm 0,96$)	71,3($\pm 1,70$)	1770($\pm 3,21$)	8,469($\pm 0,015$)	41,80($\pm 0,13$)

Table 9.7: Résultats en exécution longue

	ILS-FI				EA			
Version	durée(s)	long.	long. moy.	long. max chemin	durée(s)	long.	long. moy.	long. max chemin
min-sum	418,51	1182	5,656	28,89	316,08	1202	5,749	30,99
min-max	391,05	1326	6,347	23,50	366,97	1471	7,037	29,97
moyenne	404,8($\pm 0,23$)	1254($\pm 2,47$)	6,001($\pm 0,012$)	26,2($\pm 0,08$)	341,5($\pm 1,61$)	1336($\pm 6,09$)	6,393($\pm 0,029$)	30,48($\pm 0,15$)

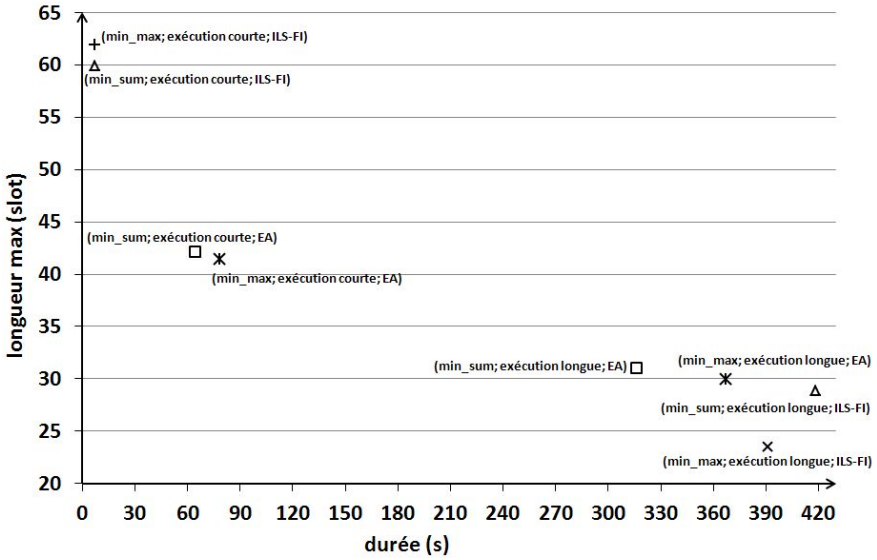


Figure 9.5: Longueur maximum pour les versions min-sum et min-max sur N4.

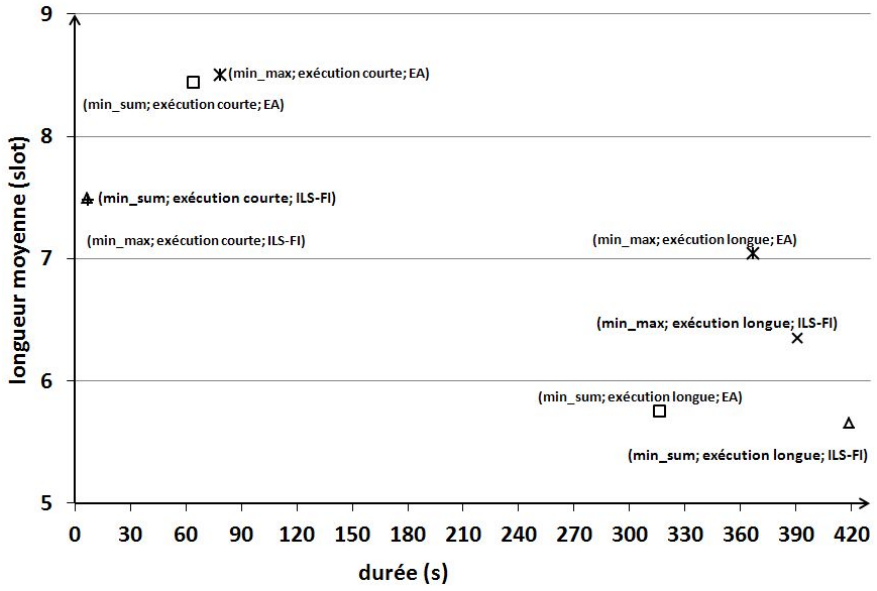


Figure 9.6: Longueur moyenne pour les versions min-sum et min-max sur N4.

9.5 Conclusion

Nous avons étendu nos évaluations expérimentales à l'approche évolutionnaire proposée. L'algorithme évolutionnaire présente une efficacité à résoudre le problème avec reconfiguration dynamique que ne possède pas la recherche locale. L'algorithme mémétique réussit sur ce cas dans une moindre mesure. Ou bien la recherche locale ne permet pas l'obtention de 100% de chemins construits, ou bien l'algorithme mémétique est-il trop lent à résoudre le problème. On peut noter toutefois un mode de fonctionnement complémentaire des méthodes. La recherche locale est adaptée à la résolution de l'instance de grande taille, sans doute du fait de son faible coût en ressource mémoire. L'algorithme évolutionnaire est lui adapté à la résolution de l'instance reconfigurable, sans doute du fait de la grande diversité des solutions générées. L'algorithme mémétique constitue un compromis entre les deux méthodes précédentes. Il est à la fois plus rapide que EA sur l'instance de grande taille, et permet de résoudre le cas reconfigurable, bien qu'avec un temps d'exécution assez long. Globalement, nous pensons avoir mis en évidence les avantages et inconvénients des différentes stratégies de recherche proposées, en vue d'apporter des réponses variées au problème de routage reconfigurable à garantie de trafic.

Conclusion générale

Les objectifs visés dans cette thèse sont principalement le développement de méthodes heuristiques et métaheuristiques appliquées aux problèmes de routage reconfigurable avec garantie de trafic dans les NoC. Nos travaux se situent dans le cadre du NoC μSpider II, qui est à la fois une architecture et une méthodologie de conception de NoC. Du point de vue de l'architecture, un NoC μSpider II peut avoir une topologie irrégulière, il comporte des interfaces réseau (NI) adaptatives, emploie un routage à la source combiné à une mémorisation *wormhole*. Il utilise principalement un multiplexage temporel pour garantir la bande passante des communications. Des tables *Time Division Multiple Access* (TDMA) d'émission sont associées aux composants *Intellectual Property* (IP) émetteurs et spécifient des dates d'émission de messages permettant de synchroniser les échanges et éviter tout conflit de paquets. L'approche de communication est dite à garantie de trafic. Accessoirement, des mécanismes de contrôle de flux à effet domino sont prévus dans la méthodologie pour gérer les conflits de paquets dans le cas du trafic *Best Effort*.

Nos travaux portent sur des problèmes à garantie de trafic uniquement. Du point de vue du flot de conception, nos travaux se situent aux étapes de spécification et de calcul des chemins de données permettant le transfert sans conflit des quantités de paquets requis par les besoins de l'application.

Nous avons présenté un problème d'optimisation combinatoire de calcul de plus courts chemins avec contraintes, qui modélise le routage sans conflit avec un trafic garanti fondé sur la technique TDMA. Nous l'avons nommé « problème cyclique des K-plus-courts chemins sans conflits » ou CKPP. Les messages sont émis de façon cyclique. Le principal but est la répartition des slots de temps disponibles pour un routage sans conflit, tout en minimisant la longueur totale des chemins de communication. Un modèle linéaire en nombres entiers (ILP) a été défini pour ce problème, mais seules les instances de très petites tailles ont été résolues de manière exacte avec succès en temps raisonnable.

Pour apporter plus de souplesse dans l'utilisation des ressources de communication, nous considérons la possibilité de reconfiguration dynamique du routage au moment de l'exécution. Dans ce cas, et ainsi que cela est prévu par la méthodologie μSpider II, plusieurs tables TDMA sont associées à un même émetteur et spécifient des chemins de communication et bandes passantes différents. Chaque émetteur peut changer de table d'émission à chaque début de cycle, et ainsi modifier dynamiquement ses chemins de communication à l'exécution. Le but est alors de permettre la réaffectation des slots de temps vacants sans perturber le système. Nous avons nommé ce problème « problème cyclique des K-plus-courts chemins reconfigurables sans conflits » ou CRKPP.

Nous avons mis en évidence le caractère NP-difficile au sens fort du problème qui est une combinaison du *Bin Packing* avec une extension d'un problème de chemins disjoints. Il s'agit d'une variante d'un problème de flot insécable, auquel il faut ajouter une contrainte d'occupation temporelle et cyclique des arcs. Cette particularité nous permet de le rapprocher des problèmes de plus courts chemins avec fenêtres de temps.

L'étude de la spécification des flots de communication et des bandes passantes, qui constituent la donnée d'entrée des problèmes de routage, nous a permis de mettre en évidence des difficultés inhérentes à la méthode de conception. La décomposition de la bande passante entre les composants IP émetteurs et récepteurs par le concepteur s'apparente à la résolution d'un problème de flot maximum non trivial. De même, le respect des contraintes d'évitement de conflit entraîne des limitations sur les possibilités de la reconfiguration dynamique. Nous avons vu que la reconfiguration de type asynchrone proposée pouvait autoriser l'utilisation à plein des récepteurs à différents moments, mais qu'il n'en était pas de même pour les émetteurs. Ceux-ci ne peuvent accroître leur bande passante que dans la limite imposée par les configurations des récepteurs associés. Permettre de modifier ces limites supposerait un changement synchrone des tables d'émission de plusieurs émetteurs simultanément, ce qui n'est pas considéré dans la méthodologie μSpider II. Dans le même temps, cette analyse des flots de communication nous a conduit à proposer un générateur d'instances aléatoires pour le CKPP et le CRKPP. Celui-ci permet de générer rapidement un grand nombre de jeux de trafic aléatoires variés, en se basant sur les instances de cas réels d'application de départ. Ces instances sont utilisées pour tester la robustesse de nos approches de résolution.

La difficulté du problème de routage nous a conduit à proposer des approches de résolution heuristiques et métaheuristiques. Nous avons utilisé la structure du graphe spatio-temporel étendu (TEG) comme base commune aux approches de résolution. Le TEG mémorise l'occupation temporelle des arcs du réseau. Une adaptation de cette structure permet de traiter le problème reconfigurable CRKPP en se basant sur une condition d'accès exclusif à un arc énoncée par le théorème (3). Nous évitons ainsi l'apparition de conflits entre paquets tout en permettant une réutilisation des slots de temps libérés durant la reconfiguration dynamique des chemins.

Nous avons conçu des opérateurs élémentaires de manipulation de chemins communs aux méthodes de recherche et fondés sur l'utilisation du TEG. La combinaison des opérateurs élémentaires dans différentes stratégies d'exploration de l'espace des solutions nous conduit à proposer trois types de méthodes de résolution. Le premier type de méthodes repose sur des recherches locales itérées qui correspondent à une utilisation poussée de l'opérateur de voisinage suivant des parcours de type glouton ou en profondeur. La difficulté des recherches locales à traiter le problème reconfigurable nous a amené à proposer un deuxième type de méthode. Nous avons choisi le schéma d'un algorithme évolutionnaire. Il permet, par la gestion d'une population de solutions, d'accroître la diversité des solutions examinées. Puis, en couplant la dynamique de diversification et de sélection de l'approche évolutionnaire avec une des recherches locales précédentes, nous avons proposé un algorithme mémétique cherchant à réunir les qualités des deux types d'approches précédentes.

L'évaluation des méthodes de recherche est réalisée par des expérimentations sur un ensemble d'instances de NoC de tailles croissantes provenant d'applications réelles, et sur des jeux de trafic aléatoires qui en sont dérivés obtenus à l'aide du générateur

d'instances aléatoires que nous avons développé. Les résultats mettent en évidence la complémentarité des composants algorithmiques de base, l'influence de la taille du voisinage et l'influence du niveau de saturation en trafic du réseau sur les performances. Par exemple, l'algorithme de plus courts chemins de type Dijkstra modifié appliqué au TEG joue un rôle déterminant dans l'amélioration très nette des performances des méthodes de recherche. En ce qui concerne les méthodes de recherche locale, nous avons constaté que seules les instances du CKPP ont pu être résolues efficacement. Nous avons observé une très bonne performance de la recherche locale dans la résolution de l'instance de grande taille, mais constaté son échec à résoudre le problème avec reconfiguration dynamique, la recherche restant confinée dans des minima locaux indésirables. L'algorithme évolutionnaire a fourni les meilleures performances dans la résolution du problème avec reconfiguration dynamique, qu'il résout avec succès. L'algorithme mémétique constitue un compromis des performances obtenues avec la recherche locale et l'algorithme évolutionnaire. En dépit du temps d'exécution important dans le cas reconfigurable, l'algorithme mémétique est nettement plus rapide que l'algorithme évolutionnaire sur l'instance de grande taille. Nous avons aussi évalué la différence de performance des algorithmes suivant la minimisation de la longueur moyenne ou maximale d'un chemin, qui correspond respectivement aux versions min-sum et min-max du problème. Nous avons pu établir une corrélation entre la configuration de l'algorithme et sa capacité à minimiser l'objectif requis correspondant. Ces expérimentations ont mis en évidence l'adéquation des approches de résolution proposées au problème de routage à garantie de trafic, à la fois standard et avec reconfiguration dynamique.

Par rapport à l'existant dans le domaine, nous pensons avoir contribué par le CKPP et le CRKPP à énoncer de manière formelle et standard des problématiques de routage le plus souvent présentées de manière informelle et succincte dans les travaux en architecture de système sur puce (SoC). Par nos propositions de modèles précis énoncés en termes de graphes et formalismes usuels en optimisation combinatoire, nous contribuons à faciliter la reproduction des méthodes de résolution proposées dans le domaine et l'évaluation comparative des algorithmes. Nous pensons avoir ainsi remédié à des insuffisances dans le domaine du routage dans le NoC. Dorénavant, nous pouvons mieux jauger la complexité des problèmes de routage dans le domaine des NoC et les relier aux problèmes de routage standard, à la fois dans les réseaux de communication et les réseaux de transports.

Nous proposons une modélisation sous la forme d'un problème de flot maximum standard des contraintes de construction de flots de données valides dans le NoC, en se basant sur l'analyse des graphes de dépendance des communications de l'application. De plus, nous contribuons à l'aide du générateur de trafics aléatoires et des jeux de trafic issus de cas réels, à proposer des jeux de tests standards, reproductibles et accessibles pour des travaux et comparaisons futures.

Dans nos travaux, nous pensons avoir proposé une meilleure délimitation et analyse des problématiques de reconfiguration. Pour la première fois, nous apportons à ces

problématiques de reconfiguration une réponse algorithmique concrète. Celle-ci étant mise en œuvre dans une architecture de NoC existante. De même, nous pensons avoir fourni des outils algorithmiques génériques aisément adaptables à différentes situations et problèmes de routage apparentés aux nôtres. Ces problèmes sont par exemple, ceux de chemins disjoints, de flots multiples insécables et de routage avec fenêtre de temps. Ils sont soit des cas particuliers ou bien des versions apparentées de notre problème pour lesquels l'utilisation d'un TEG est appropriée.

Bien que l'approche ai été conçue en tant qu'outil du NoC μspider II, des variantes et simplifications du problème peuvent être proposées dans le cadre du routage *Best-Effort*, en ne considérant par exemple que le respect des capacités des liens sans ajouter de contrainte de synchronisation. De plus, la formalisation du problème proposée en termes d'objets génériques, tels que des graphes et des plus courts chemins spatio-temporels, doit faciliter la réutilisation des outils et modèles proposés dans de nombreux contextes et d'autres NoC. Nous avons essayé de contribuer à cette démarche d'abstraction et de standardisation des problèmes de routage pour les rendre plus génériques et davantage indépendants du matériel.

Les présents travaux doivent aider à guider la construction de métaheuristiques adaptées aux problèmes de routage où le temps est primordial. Ils doivent éclairer sur l'utilisation, la portée, et la combinaison de mécanismes de reconfiguration et d'adaptation aux variations de trafic (reconfiguration asynchrone et synchrone, gestion dynamique des conflits entre paquets).

Perspectives

Les perspectives d'extension de nos travaux portent sur les constats que nous avons eu à faire à propos des limites de la reconfiguration dynamique, et sur l'évolution du nombre de nœuds et du trafic du NoC.

Ces limites ont trait à la reconfiguration des émissions de messages et au besoin d'introduire plus de souplesse encore dans l'utilisation des ressources du NoC. Plusieurs pistes apparaissent pour les surmonter.

En ce qui concerne la reconfiguration des bandes passantes par le mécanisme de permutation de tables TDMA, la limite tient à l'impossibilité d'opérer des changements synchrones des tables TDMA par plusieurs émetteurs simultanément. Introduire ce type de mécanisme de reconfiguration synchrone dans l'architecture matérielle du NoC ne semble pas une chose aisée, mais il conviendrait d'en étudier la faisabilité. En tant que problème d'optimisation combinatoire, il s'agit d'une extension du problème présenté ici dans laquelle la réutilisation des *time-slots* serait étendue par la prise en compte de groupements de tables TDMA mutuellement exclusifs. Il nous paraît intéressant d'explorer cette voie à la fois au niveau matériel et algorithmique sur des cas simples dans un premier temps.

En ce qui concerne l'introduction de plus de flexibilité dans l'utilisation des ressour-

ces du NoC, l'approche de reconfiguration par permutation des tables TDMA répond davantage à des scénarii impliquant des modifications durables, et peu fréquentes, des besoins en trafic de l'application. La garantie de bande passante autorisée par le multiplexage temporel présuppose néanmoins un sur-dimensionnement de celle-ci suivant des scénarii de trafic évalués au pire cas. Relâcher cette contrainte nous paraît envisageable et nous pensons également pouvoir appliquer nos approches suivant des scénarii de trafic évalués en moyenne. Cela nous amènerait vers la problématique de la combinaison du trafic GT et BE que nous avons évoquée dans ce document. La condition minimale à assurer lors de la construction des chemins est alors de garantir l'obtention d'un graphe de communication exempt d'interblocage. La question du partage des liens physiques entre trafic GT et BE est aussi posée. Introduire ces éléments de souplesse supplémentaires dans la gestion des ressources induit nécessairement des modifications de l'architecture matérielle du NoC μSpider II. Une forte synergie entre l'architecte du SoC et le concepteur en recherche opérationnelle est nécessaire. Nous proposons de contribuer à ces problématiques dans un avenir proche.

Nous pensons pouvoir traiter efficacement des instances de NoC de taille variable du problème. Le traitement du problème s'insère dans une démarche de conception par essai/erreur pour dimensionner le réseau suivant l'accroissement de la demande en trafic. Les recherches futures devraient contribuer à faciliter la démarche du concepteur par l'automatisation des tâches répétitives. Pour cela, des extensions du problème prenant en compte l'ajustement et la modifications des dimensions du réseau peuvent être proposées.

L'approche à garantie de trafic repose sur une hypothèse de fort synchronisme du NoC avec une horloge unique partagée entre composants et routeurs. La question se pose de l'extension des problématiques à garantie de trafic à des réseaux de grandes tailles dans lesquels la contrainte de synchronisation n'est pas applicable globalement. Différentes variantes du problème d'optimisation proposé consistent à considérer conjointement ou non la synchronisation des échanges, la possibilité de conflits entre paquets, les contraintes de capacité, la survenance de *deadlocks*. Pouvoir traiter tous ces aspects simultanément, constitue une des pistes de contribution future au routage dans les NoC.

Glossaire

Les abréviations et acronymes présents dans ce glossaire sont signalés dans le texte par un astérisque (*).

ACG Application Communication Graph

BE Best Effort

CDG Communication Dependency Graph

CKPP Cyclic K-conflict-free shortest Paths Problem

CPU Central Processing Unit

CRKPP Cyclic Reconfigurable K-conflict-free shortest Paths Problem

DSP Digitial Signal Processor

EDP K-Edge-Disjoint Shortest Paths Problem

FPGA Field-Programmable Gate Array

GLPK GNU Linear Programming Kit

GT Guaranteed Traffic

ILP Integer Linear Program

ILS-BI Iterated Local Search - Best Improvement

ILS-FI Iterated Local Search - First Improvement

IP Intellectual Property

IRS Iterated Random Search

MCU Micro Controller Unit

MPSoC multi-processor System-on-chip

MRT Maximum Reception Throughput

NI Network Interface

NoC Network on Chip

OS Operating System

OSI Open Systems Interconnection

QAP Quadratic Assigment Problem

QoS Quality of Service

RTIP Reconfigurable Traffic Injection Problem

SoC System on Chip

TDMA Time Division Multiple Access

TEG Time-expanded graph

TIP Traffic Injection Problem

TSL Traffic Saturation Level

UFP Unsplittable Flow Problem

Références

[1] A. Agarwal, C. Iskander, and R. Shankar. Survey of Network on Chip (NoC) Architectures and Contributions. *Journal of Engineering, Computing and Architecture*, 3(1):1 – 15, 2009. 5, 29

[2] G. Ascia,, V. Catania, and M. Palesi. Mapping Cores on Network-on-Chip. *International Journal of Computational Intelligence Research*, 1(2):109–126, 2005. 24

[3] L. Benini and G.D. Micheli. Networks on Chips: A New SoC Paradigm. *Computer*, 35(1):70–78, 2002. 5, 29

[4] J. Cong, C. Liu, and G. Reinman. ACES: application-specific cycle elimination and splitting for deadlock-free routing on irregular network-on-chip. In *Proceedings of the 47th Design Automation Conference, DAC '10, ACM New York, USA*, pages 443–448, 2010. 5, 16, 20, 27

[5] R. Dafali. *Conception des réseaux sur puce reconfigurables dynamiquement*. PhD thesis, Université de Bretagne Sud, France, 2011. 5, 8, 11, 12, 13, 14, 15, 18, 22, 23, 24, 39, 49, 54, 55, 87

[6] R. Dafali, J.-P. Diguet, A. Rossi, and M. Sevaux. A mixed integer linear programming model for routing messages in a noc. Document de travail, LabSTICC, 11 2009. 33

[7] W.J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference*, 2001. 5, 27, 29

[8] J. Delorme. *Méthodologie de modélisation et d'exploration d'architecture de réseau sur puce appliquée aux télécommunications*. PhD thesis, Institut National des Sciences Appliquées de Rennes, France, 2007. 13, 14, 23, 24, 26

[9] M. Desrochers and F. Soumis. A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR*, 26:191–212, 1988. 27, 39

[10] L. Devaux. *Flexible interconnection networks for dynamically reconfigurable architectures*. PhD thesis, Université de Rennes 1, sous le sceau de l'Université Européenne de Bretagne, 11 2011. 12, 13, 14

[11] H. Elmiligi, A.A. Morgan, and M.W El-Kharashi. A topology-based design methodology for Networks-on-Chip applications. In *Design and Test Workshop, 2007. IDT 2007. 2nd International*. 24

[12] S. Evain. *μSpider Environnement de Conception de Réseau sur Puce*. PhD thesis, Institut National des Sciences Appliquées de Rennes, France, 2006. 13, 14

[13] S. Evain and J.-P. Diguet. Efficient space-time noc path allocation based on mutual exclusion and pre-reservation. In *Proceedings of the 17th ACM Great Lakes symposium on VLSI, Italy*, 2007. 6, 24, 25, 27, 32

[14] L.R. Ford and D.R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433, 1958. 60

[15] M.R. Garey and D.S. Johnson. "Strong" NP-Completeness Results: Motivation, Examples, and Implications. *Journal of the Association for Computing Machinery*, 25(3):499–508, 1978. 27, 37

[16] T. F Gonzalez and D. Serena. Complexity of pairwise shortest path routing in the grid. *Theoretical Computer Science*, 326:155–185, 2004. 38

[17] K. Goossens, J. van Meerbergen, A. Peeters, and R. Wielage. Networks on Silicon: Combining Best-Effort and Guaranteed Services. In *Automation and Test in Europe Conference and Exhibition*, pages 423–425, 2002. 5, 53, 55

[18] A. Grasset. *Synthèse des interfaces de communication dans la conception des systèmes nonpuces: de la spécification à la génération automatique*. PhD thesis, Institut National Polytechnique de Grenoble, France, 1 2005. 14

[19] A. Hansson, K. Goossens, and A. Radulescu. A Unified Approach to Constrained Mapping and Routing on Network-on-Chip Architectures. In *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 75–80, 2005. 6, 24, 25, 27, 32

[20] M. Hazem. *Architectues des Réseaux sur puce pour Décodeurs Canal Multiprocesseurs*. PhD thesis, École Nationale Supérieure des Télécommunications de Bretagne et Université de Bretagne Sud, France, 12 2009. 13, 14

[21] I. Ioachim, S. Gélinas, F. Soumis, and J. Desrosiers. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31(3):193–204, 1998. 39

[22] A.A. Jerraya. *Conception de haut niveau des systèmes monopuces*. EGEM.: Série électronique et micro-électronique. Hermès science publications, 2002. 12

[23] H. Jingcao and R. Marculescu. Energy- and performance-aware mapping for regular noc architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(4):551–562, 2005. 24

[24] D.S. Johnson and L.A. McGeoch. The Traveling Salesman Problem: A Case Study in Local Optimization. In *Aarts, E.H.L. and Lenstra, J.K (eds) Local Search in Combinatorial Optimization*, pages 215–310. John Wiley and Sons, London, 1997. 63, 76, 100

[25] R.M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5:45–68, 1975. 26, 38

[26] E. Köhler, K. Langkau, and M. Skutella. Time-expanded graphs for flow-dependent transit times. In *Proceedings of the 10th Annual European Symposium on Algorithms*, ESA '02, pages 599–611, London, UK, 2002. Springer-Verlag. 7, 39, 60

[27] S.G. Kolliopoulos. Edge-disjoint paths and unsplittable flow. In *T. F. Gonzalez (ed) Handbook of Approximation Algorithms and Metaheuristics*. Chapman & Hall/CRC, 2007. 26, 38

[28] B. Korte and J. Vygen. *Combinatorial Problems in Chip Design*. Report. Forschungsinst. fur Diskrete Mathematik, 2008. 24

[29] M. E. Kreutz, L. Carro, C. A. Zeferino, and A. A. Susin. Communication architectures for System-on-Chip. In *Proceedings of the 14th symposium on Integrated circuits and systems design*, SBCCI '01, page 14, Washington, DC, USA, 2001. IEEE Computer Society. 24

[30] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A network on chip architecture and design methodology. In *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, pages 105 –112, 2002. 14

[31] A. Leroy. *Optimizing the on-chipcommunication architecture of lowpower Systems-on-Chip in DeepSub-Micron technology*. PhD thesis, Université Libre de Bruxelles, Belgique, 2006-2007. 13

[32] A. Leroy, P. Marchal, A. Shickova, F. Catthoor, F. Robert, and D. Verkest. Spatial division multiplexing: a novel approach for guaranteed throughput on NoCs. In *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, CODES+ISSS '05, pages 81–86, New York, NY, USA, 2005. ACM. 25

[33] Z. Lu and A. Jantsch. TDM virtual-circuit configuration for network-on-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(8):1021–1034, 2008. 6, 24, 25, 27, 32

[34] J.F. Lynch. The equivalence of theorem proving and the interconnection problem. *ACM SIGDA Newsletter*, 5:31–36, 1975. 38

[35] T. Marescaux, B. Bricke, P. Debacker, V. Nollet, and H. Corporaal. Dynamic time-slot allocation for QoS enabled networks on chip. In *3rd Workshop on Embedded Systems for Real-Time Multimedia*, pages 47–52, 2005. 6, 7, 24, 25, 27, 32

[36] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In *Proceedings of the conference on Design, automation and test in Europe*, 2004. 6, 24, 25, 27, 32

[37] R.H. Mohring, E. Kohler, E. Gawrilow, and B. Stenzel. Conflict-free Real-time AGV Routing. In *Operations Research Proceedings*, pages 18–24, 2004. 27, 39

[38] P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In *Handbook of metaheuristics*, pages 105–144. Kluwer Academic Publishers, Boston MA, 2003. 64, 80

[39] A. M. K. Mostéfaoui. *Architectures Flexibles pour la Validation et l'Exploration de Réseaux Sur Puce*. PhD thesis, Institut National Polytechnique de Grenoble, France, 08 2009. 13, 19

[40] S. Murali and G. De Micheli. Bandwidth-Constrained mapping of cores onto NoC architectures. In *Proceedings of the conference on Design, automation and test in Europe - Volume 2*, DATE '04, Washington, DC, USA. IEEE Computer Society. 24

[41] S. Murali and G. De Micheli. SUNMAP: a tool for automatic topology selection and generation for NoCs. In *DAC '04: Proceedings of the 41st annual Design Automation Conference*, pages 914–919, San Diego, CA, USA, 2004. ACM. 24

[42] P.P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh. Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures. *IEEE Transactions on computers*, 54(8):1025–1040, 2005. 21, 23, 41

[43] L. Pieralisi. *Modélisation de réseau de communication flexible pour les systémes monopuces*. PhD thesis, Institut National Polytechnique de Grenoble, France, 2006. 14

[44] W.B. Powell and Z.-L. Chen. A Generalized Threshold Algorithm for the Shortest Path Problem with Time Windows. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 40, 1998. 39

[45] S. Riso. *Évaluation des paramètres architecturaux des réseau sur puce*. PhD thesis, Université des Sciences et Techniques de Montpellier 2, France, 2005. 13

[46] G. Schrimpf, K. Schneider, H. Stamm-Wilbrandt, and V. Dueck. Record Breaking Optimization Results Using the Ruin and Recreate Principle. *J. of Computational Physics*, 159:139–171, 2000. 73

[47] J. Slaviša. *Architectues Reconfigurable de Système Embarqué Auto-Organisé*. PhD thesis, Université Henri Poincaré de Nancy, France, 11 2009. 13, 14, 19

[48] W.M. Spears, K.A. De Jong, T. Back, D.B. Fogel, and H. de Garis. An overview of evolutionary computation. In *Machine Learning: ECML-93, Lecture notes in computer science*, volume 667, pages 442–459, 1993. 64

[49] R. Stefan and K. Goossens. Enhancing the security of time-division-multiplexing networks-on-chip through the use of multipath routing. In *Proceedings of the 4th International Workshop on Network on Chip Architectures, ACM New York*, 2011. 6, 24, 25, 27, 32

[50] S. Taktak, J.L. Desbarbieux, and E. Encrenaz. A Tool for Automatic Detection of Deadlock in Wormhole Networks on Chip. *ACM Transactions on Design Automation of Electronic Systems*, 1(2):1–22, 2008. 27

[51] X.-T. Tran. *Méthode de Test et Conception en Vue du Test pour les Réseaux sur Puce Asynchrones : Application au Réseau ANOC*. PhD thesis, Institut National Polytechnique de Grenoble, France, 2008. 14, 26

[52] F.B. Zhan and C.E. Noon. A Comparison Between Label-Setting and Label-Correcting Algorithms for Computing One-to-One Shortest Paths. *Journal of Geographic Information and Decision Analysis*, 4(2):1–11, 2000. 69

Liste des figures

Liste des tables

Liste des Algorithmes